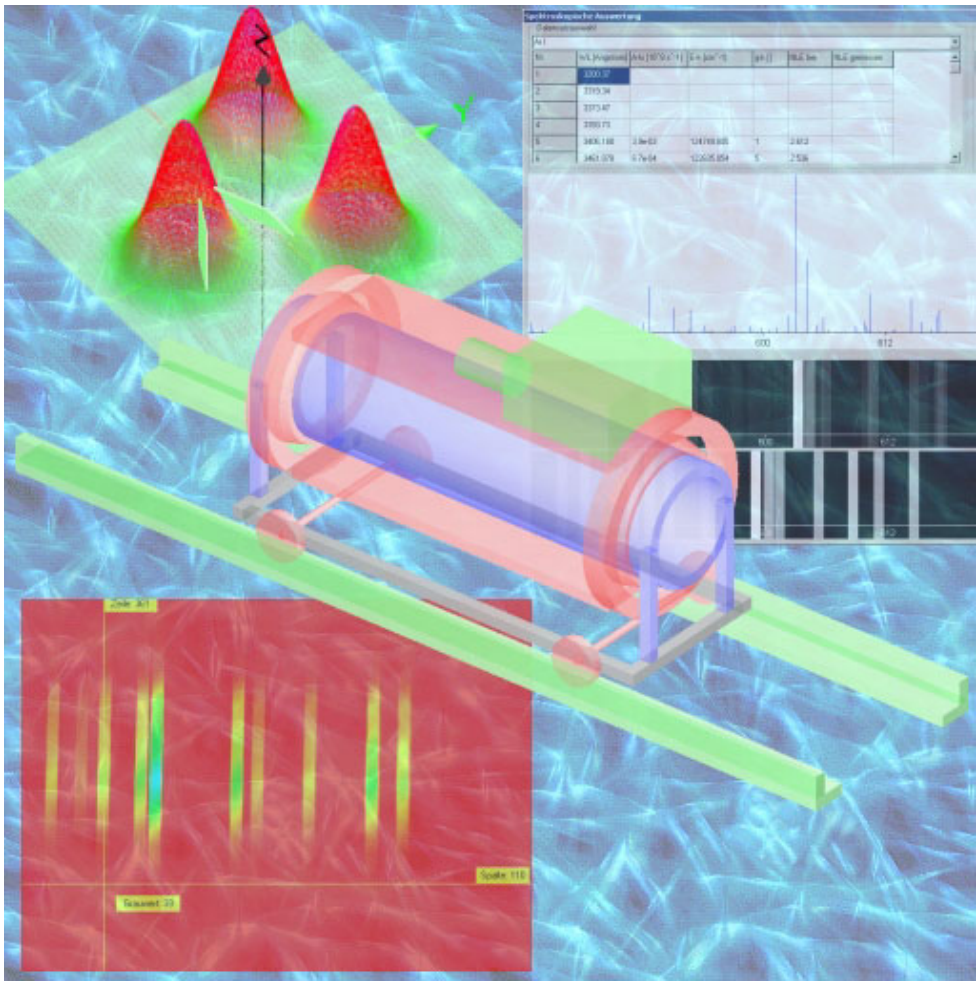
DIPLOMARBEIT

TOMOGRAFISCHE EMISSIONSSPEKTROSKOPIE AM THERMISCHEN PLASMASTRAHL



Universität der Bundeswehr München

Fakultät für Elektrotechnik und Informationstechnik

Institut für Grundgebiete der Elektrotechnik (EIT2)

2002/2003

Andreas Kraft und Björn Marlow

UNIVERSITÄT DER BUNDESWEHR MÜNCHEN

FAKULTÄT ELEKTROTECHNIK UND INFORMATIONSTECHNIK
INSTITUT FÜR GRUNDGEBIETE DER ELEKTROTECHNIK (EIT 2)

DIPLOMARBEIT

TOMOGRAFISCHE EMISSIONSSPEKTROSKOPIE AM THERMISCHEN PLASMASTRAHL

ANGEFERTIGT VON:

Björn Marlow — Andreas Kraft

MATRIKELNUMMER:

913 460 — 925 810

IM

HT 2002 / WT 2003

BETREUER:

Prof. Dr.-Ing. Klaus Landes
Dr.-Ing. Günter Forster

Unser herzlicher Dank gilt

Herrn Prof. Dr. Ing. K. Landes für die Ermöglichung der Diplomarbeit.

Herrn Dr. Ing. G. Forster für die Überlassung des interessanten Themas und die hervorragende Betreuung.

Weiterhin gilt unser Dank allen Mitarbeitern des Instituts EIT2, die uns alle auf ihre Weise bestmöglich unterstützt haben. Im Besonderen den Mitarbeitern des Labors und der Werkstatt.

Inhaltsverzeichnis

1 Einleitung 5
2 Messobjekt 8
2.1 Standardplasmabrenner vom Typ „F4“ 9
2.2 Dreikathodenplasmabrenner vom Typ „TRIPLEX“ 10
2.3 Breitstrahlplasmabrenner vom Typ „LARGE“ 11
3 Messverfahren 12
3.1 Tomografische Komponente 18
3.2 Spektroskopische Komponente 22
4 Hardware 25
4.1 Die Kamera 26
4.2 Abbildungseigenschaften der Optik 30
4.3 Die Wellenkupplung 37
4.4 Die Kühlschlange 37
5 Software 38
5.1 Das Hauptformular 41
5.2 Globale Typen, Variablen und Konstanten 45
5.2.1 Variablen 45
5.2.2 Konstanten 46
5.2.3 Typen und Sets 47
5.2.4 Erstellte Klassen und Records 47
5.3 Die Datenbank „Tomografische Emissionspektroskopie“ 51
5.4 Justage und Datenaufnahme 54
5.4.1 Das Formular „Schrittmotorsteuerung“ 54
5.4.2 Das Formular „Kamera-Livebild“ 57
5.4.3 Das Formular „Aufnahme“ 60
5.4.4 Das Formular „Drehachsen-Bestimmung“ 62
5.4.5 Das Formular „Bildbereich auswählen“ 69
5.4.6 Das Formular „Messung“ 71
5.4.7 Das Formular „Spektrale-Justage“ 81
5.4.8 Das Formular „Grafische Überwachung“ 93
5.5 Datenverarbeitung 96
5.5.1 Das Formular „Tomografische Auswertung“ 96
5.5.2 Das Formular „Serienbildbrowser“ 97
5.5.3 Das Formular „Simulation tomografischer Messdaten“ 99
5.5.4 Das Formular „Spektroskopische Auswertung“ 103
5.5.5 Das Formular „3D Browser“ 106
5.6 Zusätzliche Hilftsformulare 110
5.6.1 Das Formular „Messdatenaufarbeitung“ 110
5.6.2 Das Formular „Checklist“ 113
5.6.3 Das Formular „Boltzmann – Plot“ 117
5.6.4 Das Formular „HTML – Konverter“ 118
5.6.5 Das Formular „Messwerte addieren“ 119
6 Messungen und Ergebnisse 120
7 Zusammenfassung 135
8 Literaturverzeichnis 136

1 Einleitung

Ziel der vorliegenden Arbeit war die Inbetriebnahme eines vorhandenen Versuchsstandes zur tomografischen Emissionsspektroskopie sowie die Durchführung von Messungen an institutseigenen Plasmaquellen.
Plasma ist ein ionisiertes Gas. Es stellt einen eigenen Aggregatszustand der Materie dar. In diesem existieren neben Neutralteilchen auch positive und negative Ladungsträger in gleicher Konzentration. Es ist deshalb elektrisch leitend und durch Magnetfelder beeinflussbar, nach außen jedoch neutral. In der Natur findet man Plasmen vor allem in der Ionosphäre und bei Blitzentladungen. Der Sonnenwind sowie ein Grossteil der Sonne sind im Plasmazustand. Technisch erzeugte Plasmen finden wir in der Beleuchtungstechnik (Nieder- und Hochdrucklampen), in der Beschichtungs- und Schneidetechnik, wie auch bei der in der Entwicklung befindlichen Kernfusion.
In der Diplomarbeit werden ausschließlich thermische Plasmen untersucht. Diese erreichen Temperaturen von bis zu 20000° C. Eine Untersuchung ist somit nicht mehr auf herkömmliche Weise möglich. Deshalb wird untersucht, ob mit tomografischer Emissionsspektroskopie, ähnlich der in der Medizin eingesetzten Computertomografie, eine Aussage über die Temperaturverteilung im Plasmastrahl gemacht werden kann.
Im Laufe der vergangenen Jahre wurden bereits einige Diplomarbeiten zu diesem Thema durchgeführt. Die Zusammenstellung zeigt folgende Tabelle:

Name, Zeit	Thema	Kurzzusammenfassung
Bernd Kohler Okt. 88 - April 89	Untersuchungen am Plasmastrahl mittels Emissionsspektroskopie	Aufnahme integraler Emissionsspektren eines Plasmastrahls (F4-Brenner) Vorteil: ➢ einfacher, kompakter experimenteller Aufbau Nachteil: ➢ Verfahren liefert nur integrale Messwerte und Ergebnisse
Andreas Bappert Okt. 94 - April 95	Tomografische Bildverarbeitung zur lokalen Bestimmung der Emissions-Koeffizienten von Plasmastrahlen	Programmierung eines Tomografie–Algorithmus (Programmiersprache: C auf DOS-Ebene) Vorteil: ➢ erster funktionsfähiger Tomografie-Algorithmus Nachteil: ➢ Speicherbegrenzung (640 KB) (Auflösung 8x8 wird erreicht)

Siegfried Pock Okt. 96 - April 97	Bestimmung der Intensitätsverteilung von Plasmastrahlen mittels Computer-Tomographie	Erste tomografische Untersuchungen am TRIPLEX-Plasmabrenner (ohne spektrale Auflösung); Auswerteprogramm in PASCAL (DOS-Protected-mode, dadurch keine Speicherbegrenzung) Vorteil: ➢ Auflösung 60x60 wird erreicht Nachteil: ➢ Detektionsapparatur ruht, Messobjekt wird gedreht
Norbert Schindler Okt. 97 - April 98	Entwurf und Aufbau eines Messplatzes zur tomografischen Diagnostik an Plasmastrahlen	➢ Computergesteuerte Tomografie-Trommel ➢ Aufbau der Hardware ➢ Konstruktion des Messplatzes
Robert Schwinn Okt. 97 - April 98	Weiterentwicklung von Rechenprogrammen der Plasmadiagnostik	➢ Das Programm NOVOTRON ➢ Entwicklung und Anleitung ➢ Software-Aufgabe: Optimierung des Tomografiealgorithmus
Jochen Marks Okt. 98 - April 99	Tomografische Emissionsspektroskopie – Realisierung der Optik und Entwicklung der Software zur Steuerung des Messplatzes	➢ Entwurfszeichnungen (Optik, Profilschiene, ..) ➢ Überblick über die für die Durchführung der Messung benötigte Software ➢ Beschreibung der Steuerungssoftware NOVOTOM ➢ Verkabelung der für die Messung notwendigen Geräte und Bauteile
Ahoyo Maxime April 99 - Dez. 00	Tomographische Emissionsspektroskopie Inbetriebnahme eines neuentwickelten Versuchsstandes	➢ Erstellung Steuerungs-, Auswerte- und Simulationsprogramms TOMO ➢ Durchführung erster Messungen

Aufbauend auf bereits vorhandenen Erkenntnissen wird eine neue Steuerungs- und Auswertesoftware „AKINA“ entwickelt. Diese ist in drei Hauptprogrammteile aufgespaltet:

- Datenaufnahme
- Datenauswertung
- Datenpräsentation

Dabei ist ein modularer Aufbau wichtig, um bei Bedarf das Programm weiterentwickeln zu können. Zur Speicherung der Messdaten wird eine XML - Datenbank genutzt. Sie ist auch ohne funktionierende Software lesbar und kann somit bei Bedarf auch ohne Hilfe des Computers ausgewertet werden.
Die Hardware wurde ebenfalls angepasst. Es wurde eine neue Kamera mit einer Auflösung von 640 x 480 Bildpunkten bei einer Auflösung von 12 Bit (Graustufen) eingesetzt. Dabei wurde gleichzeitig eine neue Halterung entwickelt, um die Kamera zu

fixieren und trotzdem eine Justierung zu ermöglichen. Damit auch längere Experimente ermöglicht werden, wird eine Kühlschlange eingebaut. Sonstige konstruktive Verbesserungen wie z.B. eine Verbesserung der Achskupplung wurden ebenfalls nötig.

Im Kapitel 2 werden die Messobjekte (thermische Plasmastrahlen) vorgestellt, deren Untersuchung das Ziel der in der Arbeit beschriebenen Messapparatur ist.
Im anschließenden Kapitel 3 sind die theoretischen Grundlagen und die Funktion der tomografischen Emissionsspektroskopie zusammengestellt. Außerdem enthält dieser Abschnitt eine Einordnung des Verfahrens in die Messtechnikpalette des Instituts.
Kapitel 4 fasst den Hardware – Aufbau des Experimentes zusammen, wobei lediglich Neuerungen und Verbesserungen ausführlich beschrieben sind.
Den zentralen Punkt dieser Arbeit stellt Kapitel 5 dar. Hier ist detailliert das Software – Paket beschrieben, mit dem sowohl die gesamte Experimentsteuerung als auch die Messdatenauswertung und Ergebnispräsentation abgewickelt wird.
Schließlich sind in Kapitel 6 noch einige Messungen zusammengestellt, die das Potential der vorgestellten Messapparatur erahnen lassen.

2 Messobjekt

Den verwendeten Plasmabrennern liegt ein gemeinsames Funktionsprinzip zu Grunde. Abbildung 2-1 zeigt dieses.

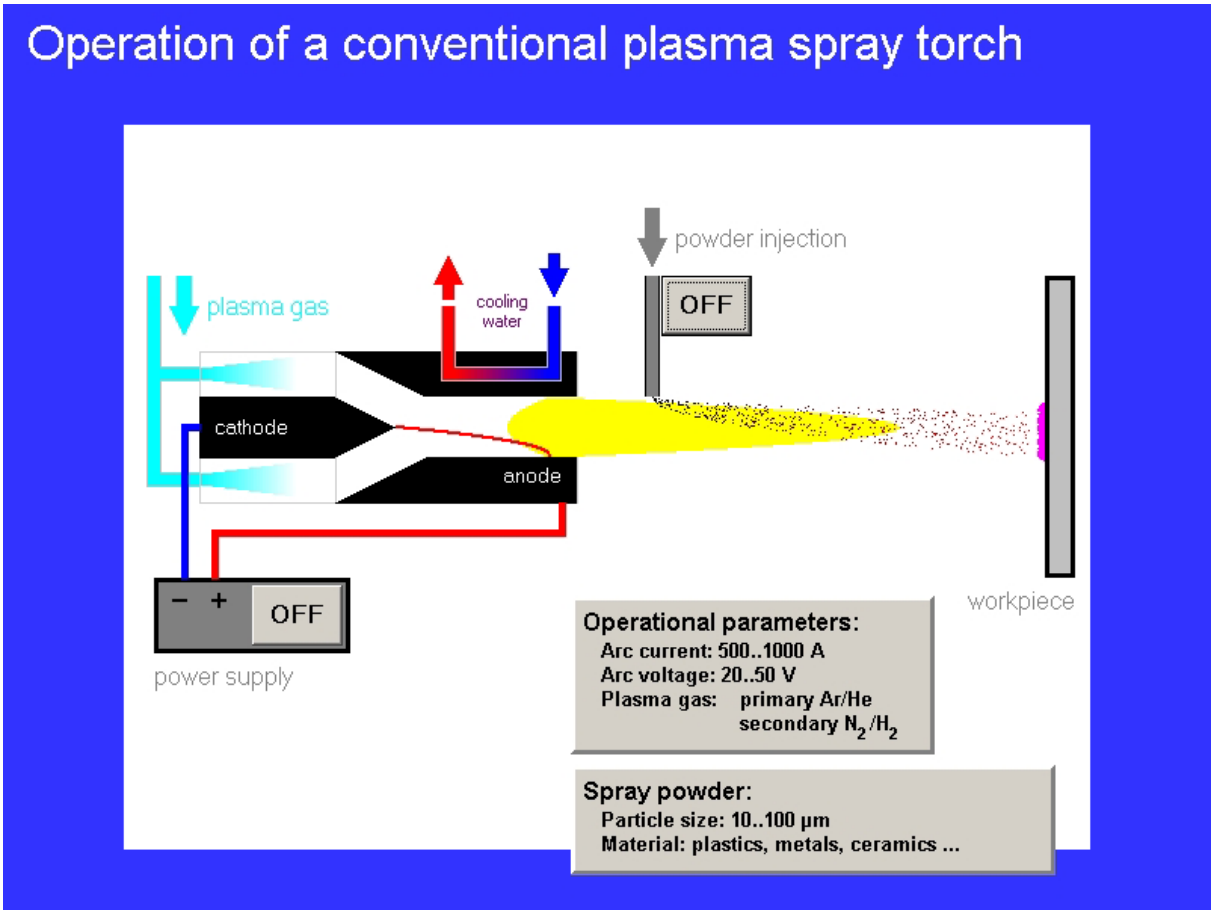


Abbildung 2-1: Funktionsprinzip Plasmabrenner

Die Leistungseinspeisung erfolgt elektrisch über einen stationären, stromgeprägten Lichtbogen. Kaltes Gas durchquert den Lichtbogen und wird zum Plasma aufgeheizt. Das daraus resultierende Stoßdominierte thermische Plasma befindet sich im thermischen Gleichgewicht bei einer einheitlichen Plasmatemperatur von einigen 1000° K bis wenigen 10000° K.

2.1 Standardplasmabrenner vom Typ „F4“

Eines der Arbeitspferde in der Industrie ist der Plasmabrenner vom Typ F4. Eine typische Elektrodengeometrie zeigt die folgende Abbildung.

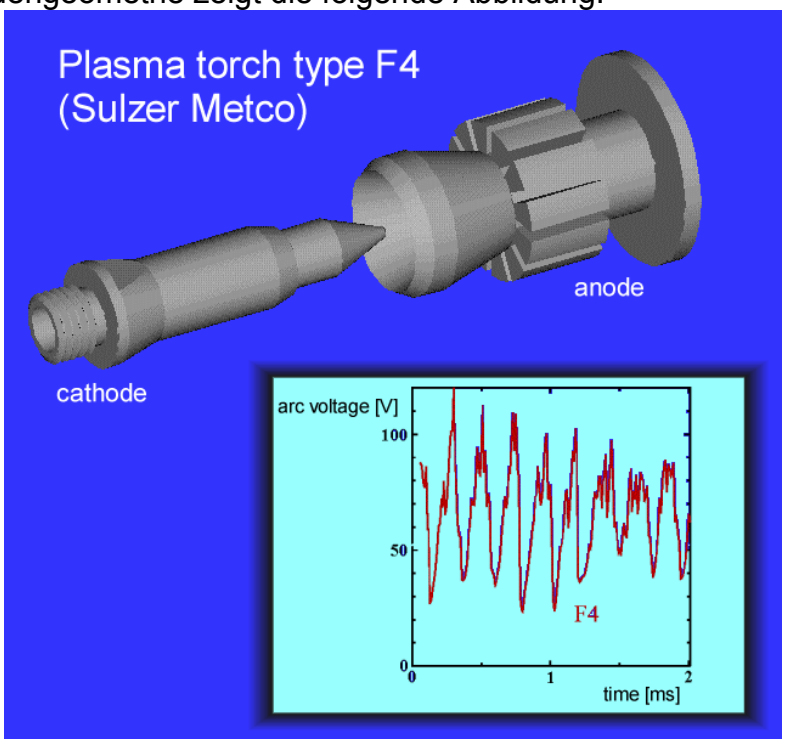


Abbildung 2-2: Plasmabrenner Typ "F4" (Sulzer Metco)

Die Kathode sowie die Anode bestehen aus Kupfer, die Kathodenspitze und die Anodeninnenwand aus hochschmelzendem Wolfram. Auf diese Weise können die hohen thermischen Belastungen durch den Lichtbogenfußpunkt an der Kathodenspitze und Anodeninnenwand bei gleichzeitiger guter Leitfähigkeit bewältigt werden. Die Form der Anode ist nach dem Prinzip der Oberflächenvergrößerung aufgebaut, um die entstehende Wärme möglichst schnell an das Kühlwasser abzugeben. Das Spritzpulver kann bei dieser Art von Plasmabrenner direkt nach der Anodenöffnung in den Rekombinationsstrahl von oben zugegeben werden. Es wird dann durch die Plasmaströmung mitgerissen, aufgeschmolzen und auf das zu beschichtende Material aufgebracht.
Bei typischen Lichtbogenspannungen von 20 – 50 V sowie Lichtbogenströmen von 500 – 1000 A erreicht dieser Brenner einen Leistungsumsatz von etwa 10 – 50 kW bei Umgebungsdrücken von 10^4 – 10^6 Pa. Aus Kostengründen findet vor allem Argon bei einem Gasdurchsatz von 30 – 200 SLM als Plasmagas Anwendung. Auch Kombinationen von Argon/Helium, Argon/Stickstoff oder Argon/Wasserstoff werden verwendet. Durch die Änderung des Gases erfolgt die Anpassung des Plasmas an die technische Anwendung.
Dieser Plasmabrenner kann auf Grund des „instabilen“ Plasmafreistrahls nur bedingt als Messobjekt eingesetzt werden. Wie auch bei Tomografieverfahren in der Medizin ist es wichtig, dass „der Patient still hält“. Besser eignet sich die Weiterentwicklung in Form des Dreikathodenplasmabrenners.

2.2 Dreikathodenplasmabrenner vom Typ „TRIPLEX“

Hauptgegenstand der Untersuchungen ist der 1995 entwickelte und seit 1997 industriell eingeführte Dreikathoden – Plasmaspritzbrenner „TRIPLEX“[8].

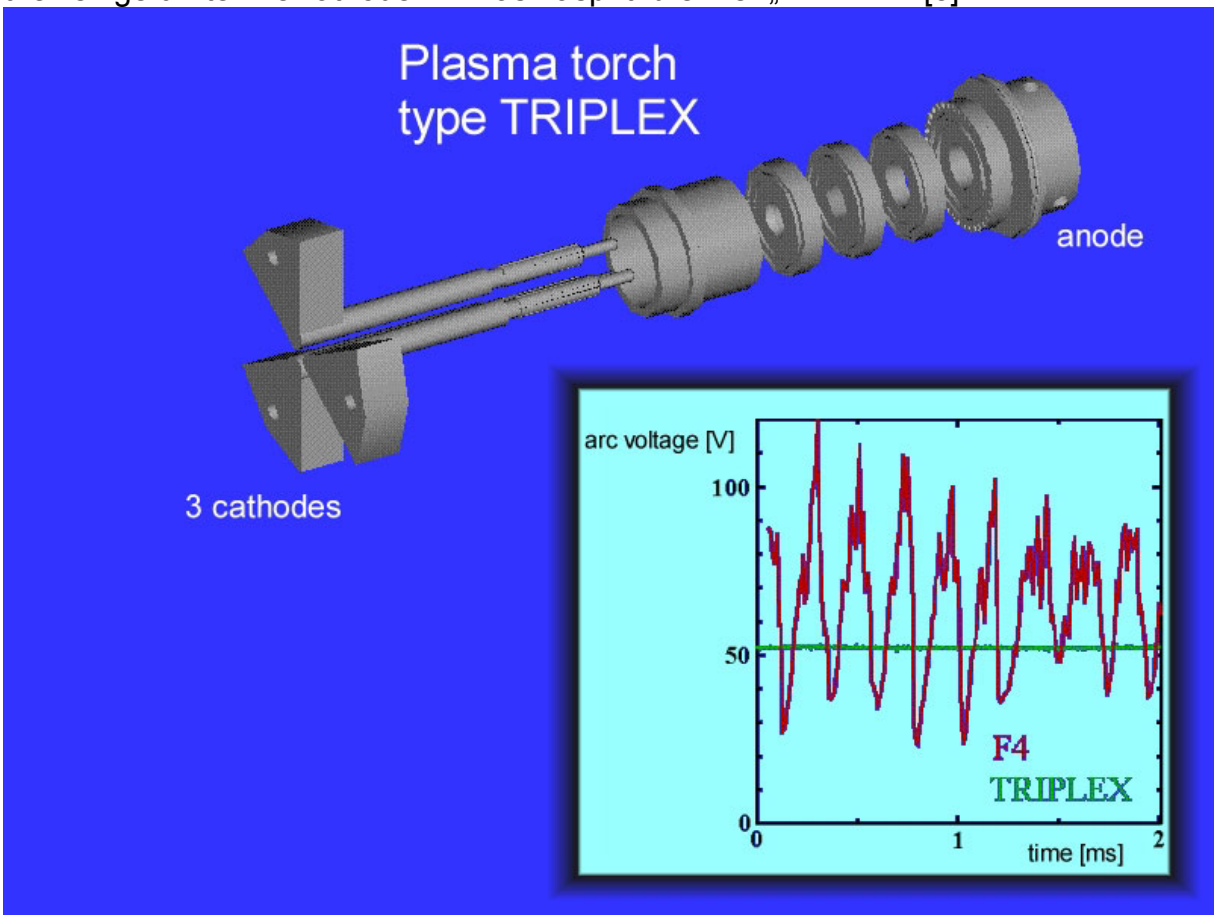


Abbildung 2-3: Plasmabrenner "Triplex" (Schema)

Im Vergleich zum herkömmlichen F4 – Brenner, der nur über eine Kathode und eine Anode verfügt, besitzt der „Triplex“ – Brenner drei Kathoden. Auf diese Weise kann der Plasmabrenner mit drei Lichtbögen betrieben werden.
Die bei F4 – Brennern auftretenden Instabilitäten durch azimutahle und axiale Wanderung des Anodenfußpunktes werden durch die Kathodenanordnung und die kaskadierte Düse des Triplex verhindert. Dies führt zu einem stabilen Plasmastrahl der vermutlich rotationssymmetrisch ist. Auch dieser Plasmabrenner wird zum Plasmaspritzen eingesetzt.
Das im Versuch verwendete Plasmagas ist reines Argon. Es wird durch den ringförmigen Gasverteiler (36x0.8 0°Grad) dem Brenner zugeführt. Nach Zündung des Plasmabrenners durch eine stromstarke Bogenentladung (ca. 1500 A), wird das Plasma durch dauernde Einspeisung elektrischer Leistung aufrechterhalten.
Vorhandene Messreihen zeigen keine ausgeprägten Wanderungen des Bogenfußpunktes und des Plasmafreistrahls [12]. Dies ist eine wesentliche Vorraussetzung für das eingesetzte Messverfahren. Deshalb wurde vor allem dieser Brenner zur Durchführung der Messungen eingesetzt.

2.3 Breitstrahlplasmabrenner vom Typ „LARGE“

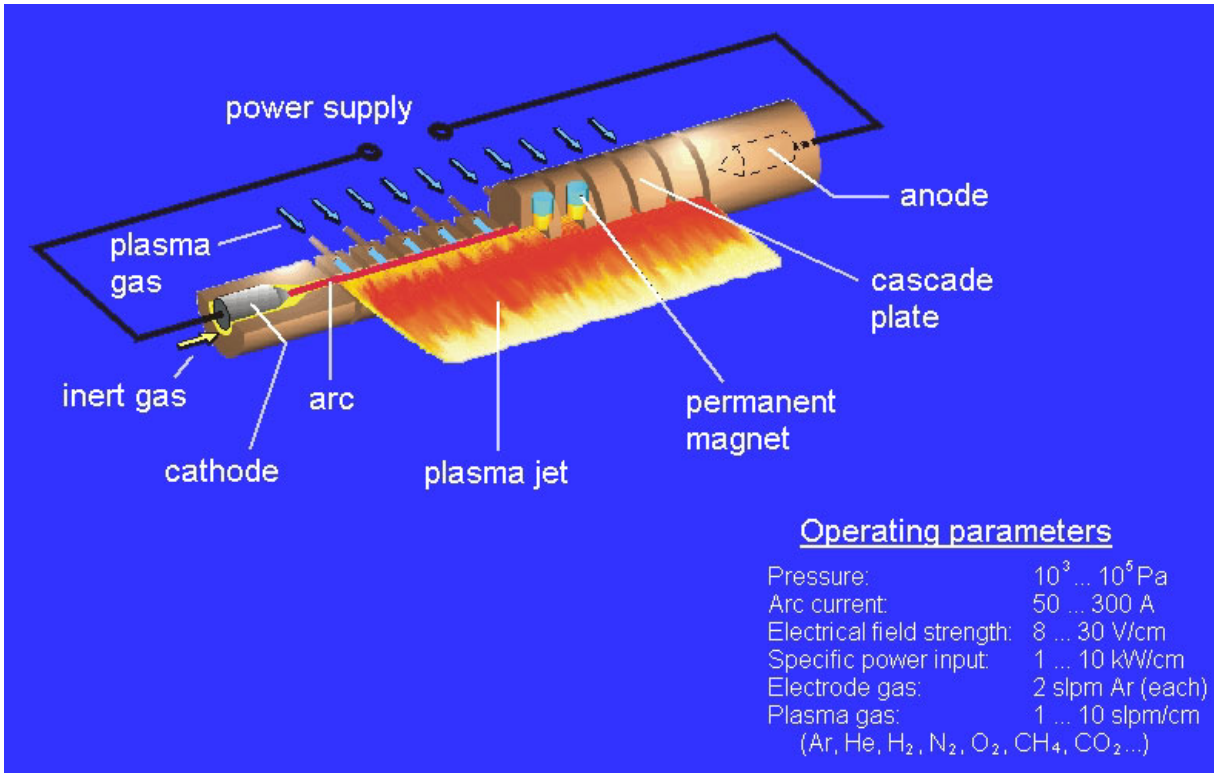


Abbildung 2-4: Breitstrahlplasmabrenner Typ "LARGE"

Der Breitstrahlbrenner vom Typ „Large“ stellt ein vollkommen anderes Brennerkonzept dar. Bei dieser Art von Brenner wird das Plasma durch zwei gegenüberliegende Elektroden im Zentralbereich des Brenners erzeugt. Der Zentralbereich des Brenners wird durch einen Stapel von gegeneinander isolierten Neutrodenplatten gebildet. Auf diese Weise kann die Brennerlänge in bestimmten Grenzen variiert werden. Nachdem der Lichtbogen durch ein elektronisches Verfahren gezündet wurde wird der Plasmastrahl durch Permanentmagnete fixiert, um eine Auslenkung auf Grund des anströmenden Plasmagases und somit eine Löschung des Lichtbogens zu verhindern.
Die Anwendungsgebiete liegen in der großflächigen Oberflächenbehandlung und PECVD. Die Breite des Plasmastrahls sorgt für eine schnelle, gleichmäßige und somit kostengünstige Behandlung großer Bauteile.
Das in dieser Diplomarbeit eingesetzte Diagnostikverfahren ist, theoretisch auch für diesen Plasmabrenner einsetzbar, da es sich hier ebenfalls um ein thermisches Plasma handelt. Die Messapparatur ist allerdings von ihren Ausmessungen nicht geeignet, um einen Plasmastrahl dieser Breite umfahren zu können. Ohne eine aufwendige Neukonstruktion der Messanordnung können deshalb keine Messungen durchgeführt werden.

3 Messverfahren

Abbildung 3-1 zeigt schematisch den Ablauf des Plasmaspritz-Prozesses. Die Aktivitäten des Institutes beschränken sich darin auf die Themenbereiche „Plasmabrenner“, „Plasmastrahl“ und „Partikelstrahl“. Für diese 3 Themenbereiche sind im Laufe der zurückliegenden Jahre eine ganze Reihe von Messtechniken und Untersuchungsverfahren adaptiert bzw. neu entwickelt worden. In den anschließenden Abschnitten sind die Schwerpunkte der im Einsatz befindlichen Diagnostik-Palette zusammengestellt, wobei jeweils ein Bezug zur tomografischen Emissionsspektroskopie hergestellt wird.

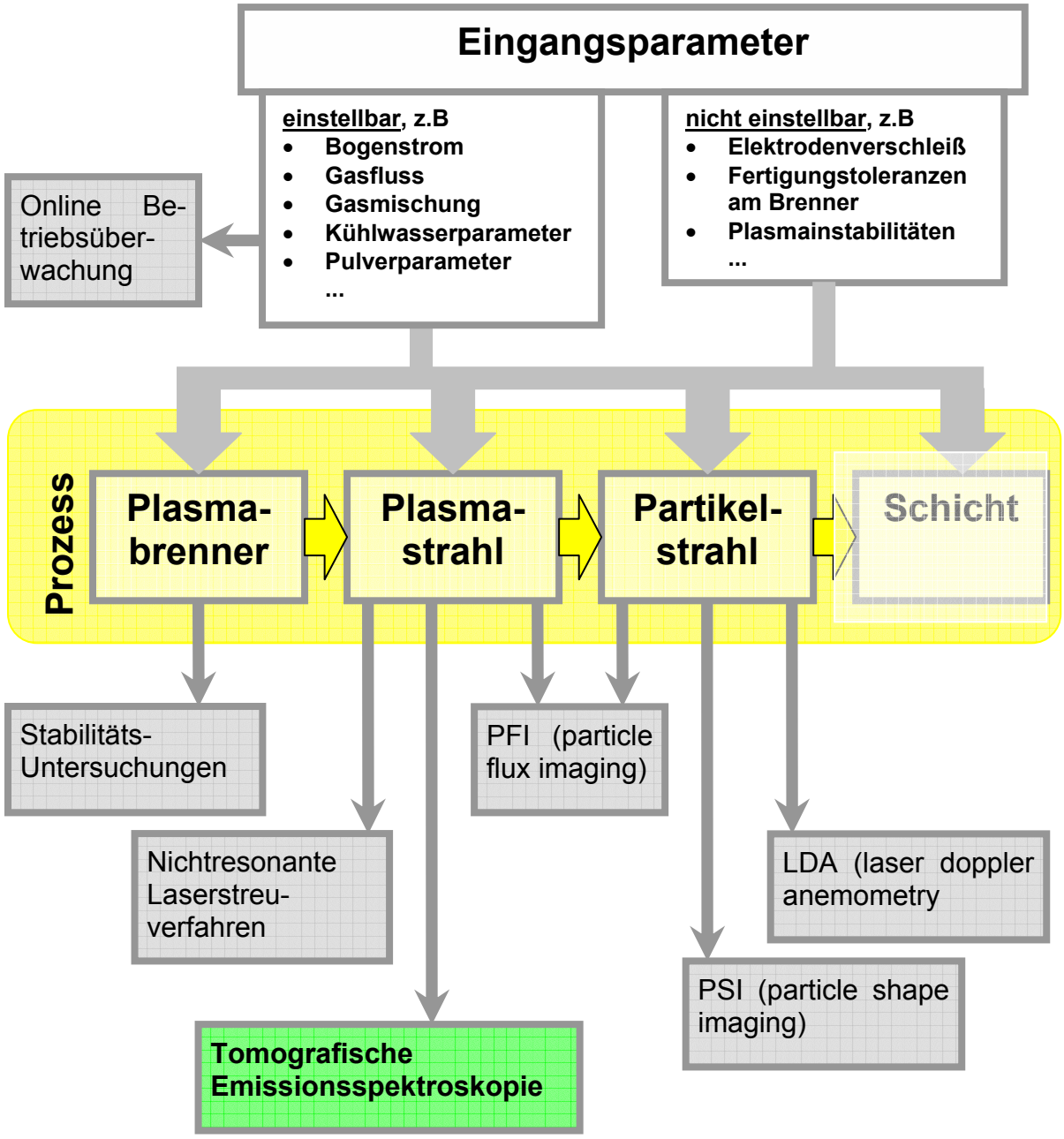


Abbildung 3-1: Diagnostikpalette rund um den Plasmaspritzprozess

Online Betriebsüberwachung

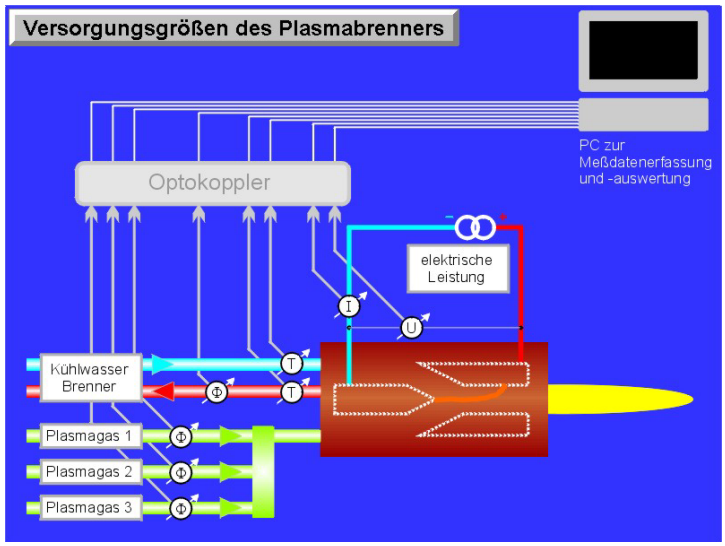


Abbildung 3-2: Überwachung der Plasmabrenner-Betriebsgrößen

Ein schematischer, aufs Wesentliche reduzierter Aufbau zur Online-Überwachung der Betriebsgrößen beim Plasmaspritzprozess ist in Abbildung 3-2 dargestellt. Im Sekundentakt werden Plasmagasdurchflüsse, Kühlwasserdurchfluss und Temperatur, und die elektrischen Größen des Lichtbogens (Bogenstrom und –spannung) aufgezeichnet [20, 21]. Bereits diese relativ einfachen Messtechniken geben Aufschluss über das Langzeitbetriebsverhalten eines Plasmabrenners. So lässt sich beispielsweise ein Verschleiß der Brennerelektroden relativ einfach an Lichtbogenstrom bzw. –spannung erkennen.

Stabilitäts-Untersuchungen

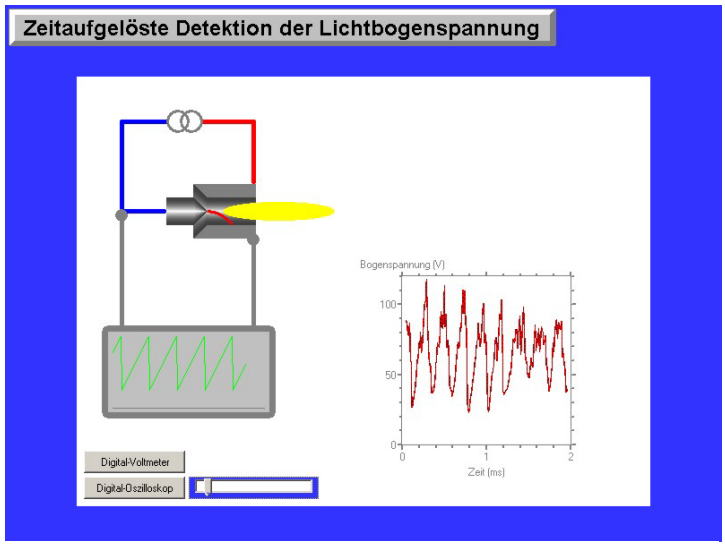


Abbildung 3-3: Stabilitätsuntersuchung Lichtbogenspannung

Detailliert untersucht wurden am Institut die zeitlichen und räumlichen Instabilitäten des Lichtbogens. Der kathodische Bogenfußpunkt ist aufgrund der Geometrie stabil auf die Kathodenspitze fixiert. Der anodische Lichtbogenfußpunkt dagegen hat beim konventionellen Plasmabrenner (z.B. F4-Brenner) zwei Freiheitsgrade. Er kann in der zylindrisch geformten Anode sowohl axial als auch azimuthal wandern.
Die axiale Bogenfußpunktswanderung führt aufgrund der Variation der Lichtbogenlänge automatisch zu Spannungsschwankungen (bei eingeprägtem Strom) und damit zu Leistungsschwankungen im Lichtbogen. Wie in Abbildung 3-3 dargestellt können diese Spannungsschwankungen sehr ausgeprägt sein, was natürlich zu einem instabilen, stark flackernden Plasmastrahl führt [8].

Diese anodischen Lichtbogenfußpunktswanderungen lassen sich auch durch eine simple Schallemissionsmessung detektieren und bewerten. Wie Abbildung 3-4 zeigt besitzt das Schallemissionsspektum ausgeprägte Maxima im Bereich weniger kHz

und korreliert eng mit dem Spektrum der Bogenspannung. Die außerordenlich hohe Lärmentwicklung eines konventionellen Plasmabrenners im Beschichtungsbetrieb (beim F4-Brenner bis zu 130 dB(A)) resultiert also vornehmlich aus der axialen Wanderung des anodischen Lichtbogenfußpunktes.

Eine weitere Stabilitätsuntersuchung wurde mit Hilfe einer Hochgeschwindigkeitskamera (12000 Bilder/sec.) durchgeführt. Abbildung 3-5 zeigt deutlich, dass sich die variierende Lichtbogenspannung und damit die variierende Leistungsaufnahme im Lichtbogen direkt in der Ausprägung des Plasmafreistrahles niederschlägt.

Diese Ergebnisse zeigen, dass sich der Plasmastrahl eines konventionellen Plasmabrenners nur bedingt für eine tomografische Untersuchung eignet. Bei dieser Messtechnik ist ein für wenigstens einige Sekunden stabiles Messobjekt nötig. Allerdings lässt sich eine Art mittlere Plasmastrahlausprägung tomografisch rekonstruieren, wenn für die Belichtungszeit der tomografischen Einzelaufnahmen ein ausreichend großes Zeitintervall gewählt wird.

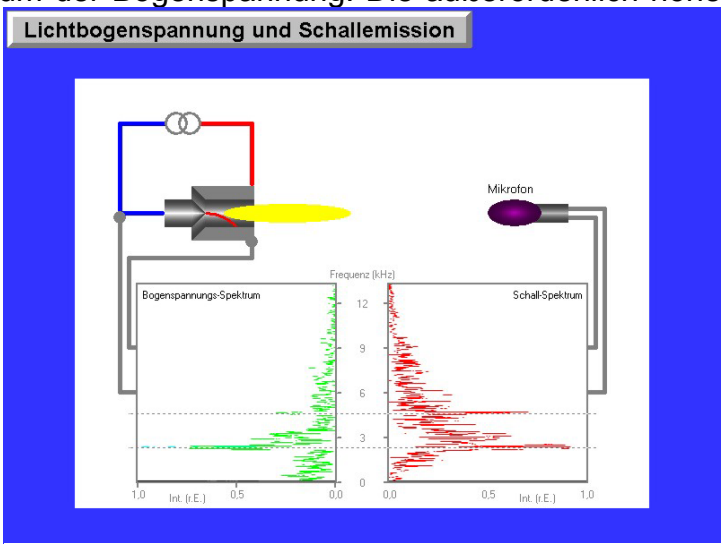


Abbildung 3-4: Stabilitätsuntersuchung Schallemission

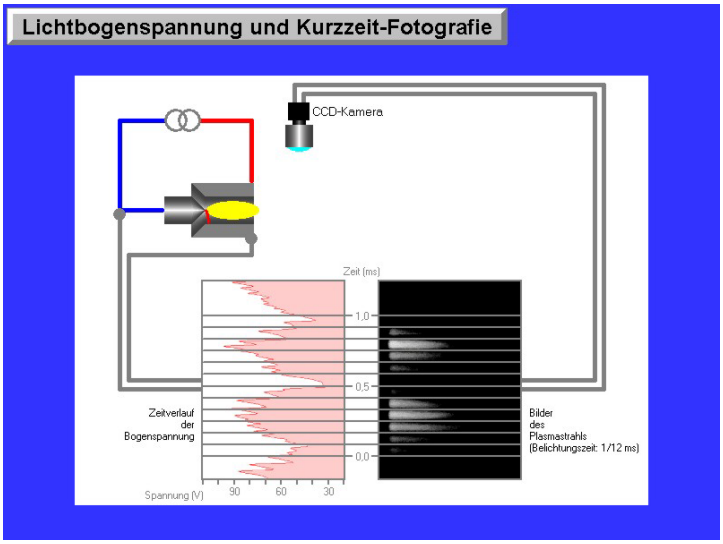


Abbildung 3-5: Stabilitätsuntersuchung Kurzzeitfotografie

Nichtresonante Laserstreuverfahren

Seit einigen Jahren wird am Institut ein Plasmadiagnostik-Verfahren entwickelt, das auf dem physikalischen Prinzip der Streuung elektromagnetischer Wellen im sichtbaren Bereich an den einzelnen Plasmagasteilchen beruht (Abbildung 3-6). Ein Hochleistungs-Pulslaser regt einen lokalen Bereich des Plasmafreistrahles an. Das bei der Abregung emittierte Streulicht wird mit hoher Empfindlichkeit und spektral aufgelöst detektiert. Aus Intensität und spektraler Verteilung des Streulichts lassen sich anschließend Rückschlüsse auf das streuende Medium ziehen [16].
Der experimentelle Aufwand für dieses Meßverfahren ist relativ hoch. Sein Einsatz ist trotzdem gerechtfertigt aufgrund der detaillierten Informationen, die vom Plasma ge-

wonnen werden. Es lassen sich mit beschriebenem Meßverfahren Anzahldichten und Temperaturen von freien Elektronen und Ionen bzw. Atomen bei hoher lokaler Auflösung ermitteln. Eine Aussage über die Gültigkeit des lokalen thermischen Gleichgewichts (LTG) im Plasmafreistrahl ist also damit möglich. Gerade die Annahme des LTG ist aber eine Grundvoraussetzung, um durch tomografische Emissionsspektroskopie zu korrekten Plasmagasparametern zu gelangen. Liegt allerdings LTG im Plasmastrahl vor, dann lassen sich mittels tomografischer Emissionsspektroskopie alle interessanten Plasmaparameter sehr viel schneller und einfacher bestimmen als über ein aufwändiges Laserstreuexperiment. Das tomografische Emissionsspektroskopie-Experiment stellt also für das vorhandene Laserstreuexperiment keine Konkurrenz, vielmehr eine vielversprechende Ergänzung dar.

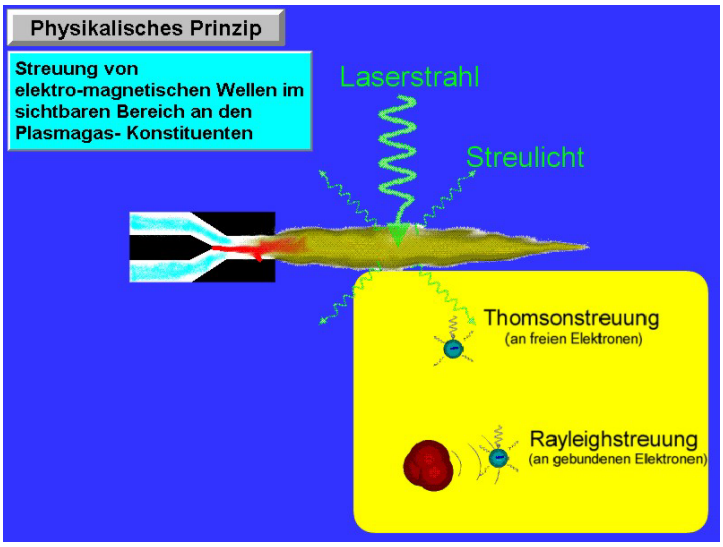


Abbildung 3-6: Nichtresonante Laserstreuung zur Bestimmung lokaler Dichten und Temperaturen der Plasmagaskonstituenten

LDA (laser doppler anemometry

Ein gängiges, zur industriellen Prozessoptimierung bereits mehrfach eingesetztes Diagnostikverfahren zur Untersuchung der Beschichtungspulverpartikel im Plasmastrahl ist die **L**aser **D**oppler **A**nemometry (LDA). Am Institut sind zwei unabhängige, mobile Systeme aufgebaut, die an die speziellen Anforderungen beim thermischen Beschichten angepasst sind. Beim LDA-Meßverfahren wird ein Laserstrahl in zwei kohärente Teilstrahlen zerlegt, die sich unter definierten geometrischen Verhältnissen am Meßpunkt im Plasmastrahl kreuzen (Abbildung 3-7). Das sich im Kreuzungspunkt der beiden Teilstrahlen ausbildende Interferenzstreifenmuster stellt das Meßvolumen dar. Durchquert ein Pulverpartikel dieses Interferenzstreifenmuster, so verursacht dieser einen modulierten Streulichtpuls (MIE-Streuung). Die Modulationsfrequenz dieses

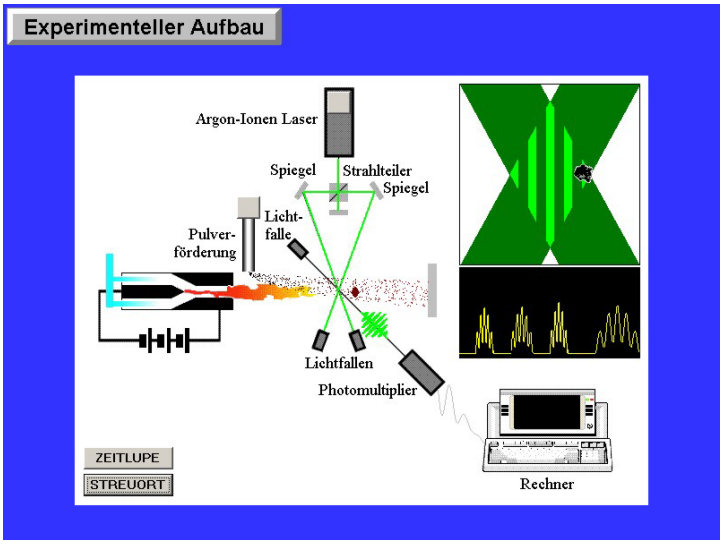


Abbildung 3-7: LDA zur Ermittlung lokaler Partikelraten und –geschwindigkeiten im Plasmastrahl

Streulichtpulses ist ein direktes Maß für die entsprechende Geschwindigkeitskomponente des Pulverpartikels [17, 18, 19].
Das LDA-Meßverfahren liefert zunächst nur lokale Meßdaten von einem Punkt im Plasmastrahl. Zur Gewinnung sehr viel aussagekräftigerer Profile muß ein Meßpunktraster abgefahren werden, was natürlich einen gewissen zeitlichen Aufwand (ca. 1 Sek./Meßpunkt) bedeutet und hohe Anforderungen an die Stabilität des Prozesses stellt.
Es ist durchaus denkbar, auch mit dem tomografischen Emissionsspektroskopieexperiment Informationen über den Partikelstrahl zu gewinnen. Die Pulverpartikel sind im Plasma auf Temperaturen geheizt, bei denen Strahlung im sichtbaren Bereich emittiert wird. Wählt man eine ausreichende Belichtungszeit für die tomografischen Einzelaufnahmen, so erscheint die von den Einzelpartikeln emittierte Strahlung als homogene Helligkeitsverteilung auf den Bildern. Die spektral aufgelösten rekonstruierten Schnittbilder könnten möglicherweise pyrometrisch ausgewertet werden; ein denkbares Ergebnis wären mittlere lokale Partikeltemperaturen.

PSI (particle shape imaging)

Abbildung 3-8: PSI zur Ermittlung von Größe und Form einzelner Pulverpartikel im Plasmastrahl

Im Rahmen einer erst kürzlich abgeschlossenen Dissertation wurde ein völlig neues Partikeldiagnostikverfahren entwickelt, **P**article **S**hape **I**maging (PSI). Dieses Meßverfahren ermöglicht es, nicht nur lokale Partikelraten und –geschwindigkeiten zu ermitteln, sondern darüber hinaus Größe und Form individueller Spritzpulverpartikel an jedem Punkt im Plasmastrahl. Das PSI-Experiment beruht auf einer schlichten fotografischen Abbildung einzelner Pulverpartikel, die Anforderungen an die einzelnen Experimentkomponenten sind allerdings extrem (Abbildung 3-8). Angefangen bei der benötigten Belichtungszeit von ca. 5ns, um „scharfe“ Momentaufnahmen der schnell fliegenden Partikel zu erhalten, über die ungewöhnliche optische Abbildung (Fernmikroskopie), um die µm-Partikel „bildfüllend“ zu projizieren, bis hin zur Lichtquelle, die das hell leuchtende Plasma überstrahlen soll, werden für das PSI-Experiment ungewöhnliche Hardwarekomponenten benötigt. Aber auch die Meßdatenverarbeitung stellt sehr hohe Anforderungen an Rechnerausstattung und Software [19].

PFI (particle flux imaging)

Das **P**article **F**lux **I**maging (PFI) Verfahren ist an sich keine Meßtechnik im herkömmlichen Sinn, mit der physikalische Parameter eines Meßobjekts ermittelt werden. Vielmehr ist es ein Verfahren, um Plasma- und Partikelstrahl insgesamt schnell und einfach zu bewerten, und um Abweichungen vom optimalen Betriebszustand zu detektieren. Der

experimentelle Aufbau des PFI-Verfahrens besteht im Wesentlichen aus einer CCD-Kamera, die im Sekundentakt Bilder vom gesamten Beschichtungsprozess aufnimmt. Ein angeschlossener Rechner ermittelt online aus den Bildern charakteristische Kenndaten und legt diese in einer Datenbank ab. Der Vergleich mit einem Referenzdatensatz erlaubt es nun, sehr schnell auf Prozessänderungen zu reagieren.

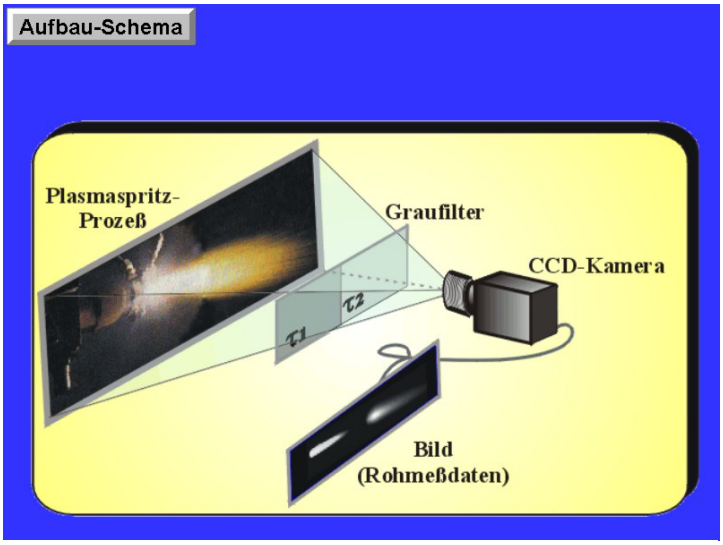


Abbildung 3-9: PFI zur Prozessüberwachung beim thermischen Beschichten

Das Verfahren hat sich inzwischen im industriellen Einsatz bewährt. Es besticht durch Robustheit und einfache Handhabbarkeit. Außerdem besitzt das PFI-Verfahren, aufgrund einer ausgefeilten Bildverarbeitung, hohe Empfindlichkeit und kann selbst geringe Prozessänderungen detektieren [8].

Diese kurze Zusammenstellung der wesentlichen Diagnostikverfahren rund um den Plasmaspritzprozess, die am Institut eingesetzt werden, soll zeigen, daß sich das tomografische Emissionsspektroskopie-Experiment harmonisch in die vorhandene Palette einreiht. In den beiden folgenden Kapiteln wird nun die physikalische Funktionsweise der tomografischen Emissionsspektroskopie, angepasst an den Plasmaspritzprozess, näher erläutert. Kap. 3.1 faßt die Grundlagen der tomografischen Rekonstruktion von Schnittbildern zusammen, während in Kap. 3.2 die spektroskopische Komponente des Meßaufbaus skizziert ist.

3.1 Tomografische Komponente

Die Methoden der Tomografie werden seit den 70er Jahren hauptsächlich in der Medizin angewendet. Sie sind ein wichtiges Hilfsmittel, um räumlich aufgelöste Messdaten zu gewinnen.
Heute hat die Tomografie einen großen Stellenwert in der Geophysik, Verfahrenstechnik und Materialforschung, gerade in Gebieten, in denen eine berührungslose, das Messobjekt nicht beeinflussende Datenaufnahme gefordert wird.
Man unterscheidet grundsätzlich zwei Arten von Tomografie: die Transmissionstomografie (Abbildung 3-10) und die Emissionstomografie (Abbildung 3-11).

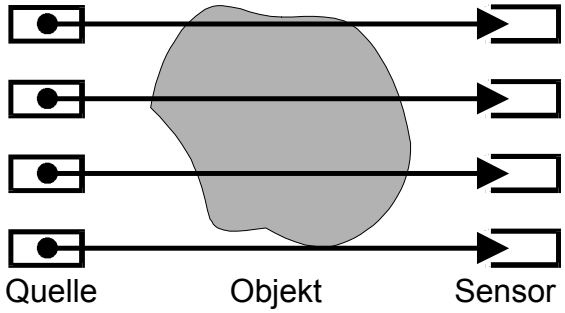


Abbildung 3-10: Prinzip der Transmissionstomografie

Die Transmissionstomografie hat sich gerade in der Medizin besonders etabliert. Das Untersuchungsobjekt wird elektromagnetischer Strahlung (z.B. Röntgenstrahlung, sichtbarem Licht, Mikrowellen), elastischen Wellen (z.B. Ultraschall) oder elektrischen bzw. statischen Feldern (z.B. Magnetfelder) bekannter Amplituden und Frequenzen ausgesetzt. Ein Empfänger misst die transmittierte, reflektierte oder gestreute Strahlung. Man erhält somit Messwerte über die Wechselwirkung der Strahlung mit dem Objekt. Die gewünschten physikalischen Größen können dann mittels tomografischer Methoden rekonstruiert werden.

Die Emissionstomografie hingegen nutzt keine externe Quelle, sondern Quellen im Inneren des zu untersuchenden Objektes. Hier können aus den Messwerten sowohl räumliche Verteilung, als auch Intensität der einzelnen Quellen ermittelt werden.

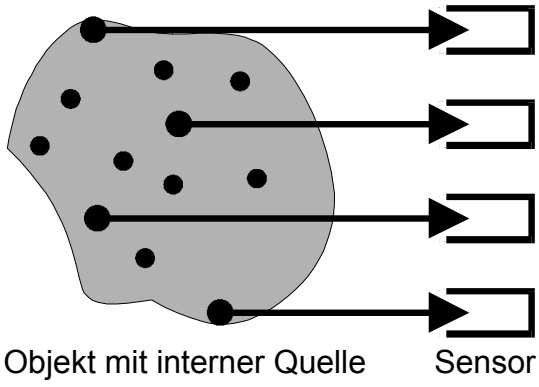


Abbildung 3-11: Prinzip der Emissionstomografie

Weiterhin unterscheidet man in der Tomografie zwischen der Untersuchung von skalaren Größen (Temperatur) oder vektoriellen Größen (Geschwindigkeit).
Obwohl die Anwendungsbreite der Tomografie sehr mannigfaltig ist, liegt ihr ein wesentliches Prinzip zugrunde:
Zur Messung kann oder soll nicht in das Messobjekt eingedrungen werden.
Die Funktion der zu ermittelnden Größen wird alleine aus den Projektionen des Messobjektes aus verschiedenen Richtungen ermittelt, wobei es keine Rolle spielt, ob dieses Objekt fest, flüssig oder gasförmig ist.

Diese Arbeit beschäftigt sich ausschließlich mit der Emissionstomografie skalarer Größen, als Objekt dient die Flamme eines Plasmabrenners.
Aufgrund der sehr hohen Temperaturen im Inneren einer solchen Flamme (ca. 20.000°K) liegt der Vorteil dieses Verfahrens -die berührungslose Messung- auf der Hand: Es gäbe keinerlei Materialien, die diesen Temperaturen längere Zeit standhalten würden.
Weiterhin wird die im Endeffekt zu messende Temperaturverteilung nicht durch kühlere Körper von außerhalb gestört.

Trotz der Vorteile birgt dieses Verfahren jedoch auch Nachteile:
Tritt die Strahlung auf dem Weg zum Empfänger in weitere Wechselwirkung (Reabsorption) mit dem zu untersuchenden Objekt, so erweitert sich das Problem von einer reinen emissionstomografischen Komponente um eine transmissionstomografische Komponente mit unbekannter Strahlungsquelle.
In unserem Fall jedoch sind die Partikel in der Flamme, die Reabsorption verursachen könnten von vernachlässigbarer Größe, so dass ein rein emissionstomografisches Problem zugrundegelegt werden kann.

Die einfachste Form der Emissionstomografie beschäftigt sich mit der Parallelprojektion:

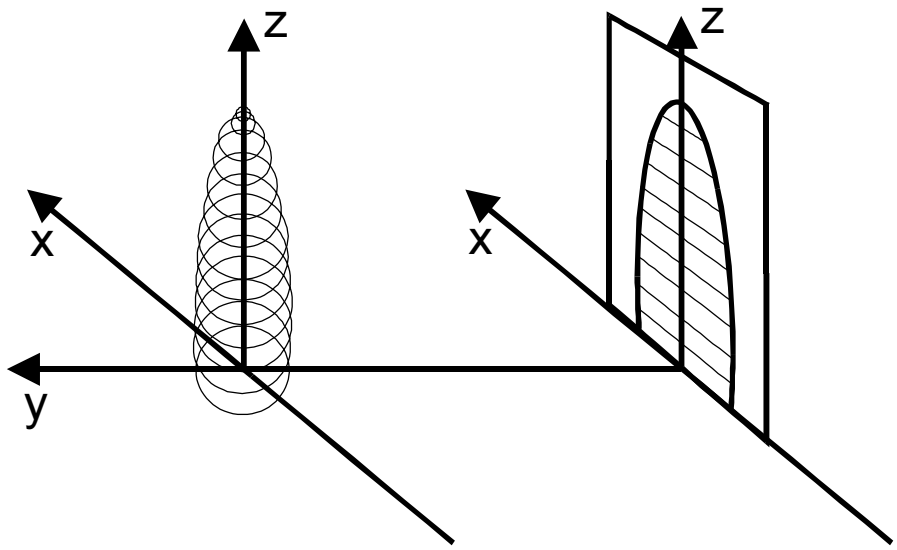


Abbildung 3-12: Parallelprojektion einer Flamme in die Kameraebene

Für die hier vorliegende Aufgabe ist es jedoch von Nöten, zusätzlich spektrale Informationen zu erhalten. Aus diesem Grund wird lediglich ein infinitesimal kleiner Schnitt (vergleichbar mit einem Ring im obigen Bild) durch die Flamme betrachtet, welcher in seine Spektralanteile zerlegt ist.
Als Folge kann jedoch nicht mehr die gesamte Flamme mit einer einzigen Aufnahme erfasst werden, sondern jede axiale Schnittebene erfordert eine eigene Aufnahme.
Aus den sich ergebenden lokalen wellenlängenabhängigen Emissionskoeffizienten $\varepsilon_\lambda(x,y,z,\lambda)$ wird dann lokal die Temperatur bestimmt.
Voraussetzung für die Anwendbarkeit des Verfahrens ist jedoch das Vorliegen des lokalen thermischen Gleichgewichts.

Wie kann von den aufgenommenen Daten auf das Messobjekt zurückgeschlosssen werden?

Dieses Problem löste J. Radon 1917 folgendermaßen:
Er zeigte, dass eine Funktion

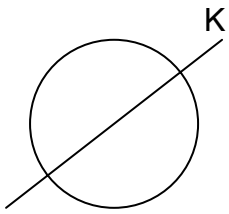
$$f : \Re^2 \to \Re$$

eindeutig aus der Gesamtheit der Integrale

$$\int_K f(x,y)ds \quad \text{K ist Gerade in } \Re^2$$

bestimmbar sein muss, wobei K hierbei durch s parametrisiert ist.

Wir betrachten also eine zweidimensionale Ebene, in der die Gerade K definiert ist. Diese zweidimensionale Ebene ist ein Schnitt durch unser Messobjekt, vergleichbar mit einem Ring aus Abbildung 3-12.



Aus dieser Betrachtung heraus entstand übrigens der Name Tomografie, der von dem griechischen τομος (=Scheibe) abstammt.
Entlang der Gerade K werden alle Werte (in unserem Fall Emissionswerte) aufsummiert. Durch Parallelverschiebung von K wird die gesamte Fläche abgedeckt. Stellt man alle aufsummierten Messwerte dar, so entsteht eine Parallelprojektion der Fläche entlang der Geraden K.

Fügt man nun noch die Richtung der geraden K, symbolisiert durch den Winkel ϕ, hinzu, so erhält man eine Schar von Projektionen $\check{f}(p,\phi)$ aus unterschiedlichen Richtungen. Dies ist das eigentliche Rekonstruktionsproblem:

$$\check{f}(p,\phi) = \int_{K_\phi} f(x,y)ds \,.$$

Nach einiger Umformung und Anwendung der Radon-Transformation ($R[f] = \check{f}$) ergibt sich eine numerisch lösbare Rekonstruktionsformel:

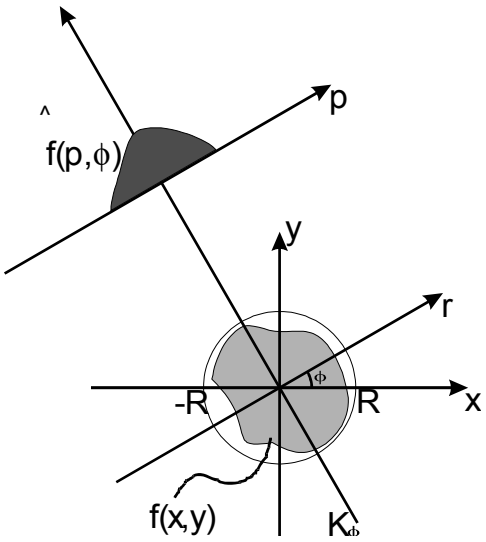


Abbildung 3-13: Projektion bei dem Winkel f entlang Kf

$$f(r,\psi) = \frac{1}{2\pi^2}\int_0^{\pi}\int_{-R}^{R} \frac{D_{(1)}\left[\check{f}\right](p,\phi)}{r\cos(\phi-\psi)-p}\,dp\,d\phi \,.$$

f ist hierbei in Polarkoordinaten angegeben, $D_{(1)}$ bezeichne die Ableitung der Radontransformierten $\check{f}(p,\phi)$ in Richtung der ersten Variable.

In unserem Fall sind die Funktionen allerdings nicht stetig gegeben, sondern technisch bedingt durch die CCD-Kamera diskret, weshalb einige zusätzliche Überlegungen getroffen werden müssen.

Anpassung der Rekonstruktionsformel auf die lokalen Gegebenheiten

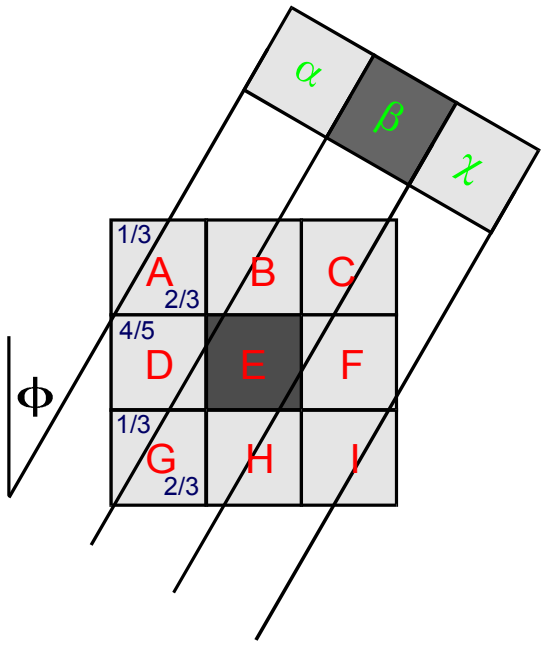


Abbildung 3-14: Lösung des Problems mit diskreten Werten

Die CCD-Kamera kann keine stetigen Helligkeitswerte aufnehmen, sondern lediglich diskretisierte Werte, abhängig von ihrer Auflösung.
Darum werden die beiden Integrale aus der Transformationsformel durch Summen ersetzt. Nun tritt jedoch ein Problem auf, zu dessen Erklärung ich den Vorgang der Transformation umdrehen möchte:
Es sei „die Scheibe“ als Lösung der Transformation bereits bekannt (siehe 3x3-Matrix in Abbildung 3-14). Die Kamera habe eine Auflösung von vertikal drei Pixeln, sie betrachte das Objekt aus einem Winkel von ϕ=30°. Welche Helligkeitswerte werden in den drei Pixeln detektiert? Hierzu müssen die neun Pixel abhängig vom Betrachtungswinkel der Kamera in Flächenfaktoren aufgeteilt werden.

Pixel α detektiert demnach $\frac{2}{3} * A + \frac{1}{10} * E + \frac{4}{5} * D + \frac{1}{3} * G$ (Alle Werte sind zum besseren Verständnis grob abgeschätzt). Ebenso verfährt man mit den Pixeln β und χ.
Realisiert ist dies in der `unit Tomografie`.
Es wird schnell deutlich, dass der Rechenaufwand exponentiell mit der aufzunehmenden Auflösung ansteigt!
Als Beispiel: Es wurde mit der Messwertesimulation ein M-Profil in den Auflösungen 60, 90, 120, 150 und 180 simuliert.

	Auflösung	60	90	120	150	180
Speicherbedarf [Byte]	vorher	6.400	6.427	6.596	6.672	6.936
	nachher	15.760	40.124	80.980	157.768	258.104
Speicherbedarf durch Alg:		9.360	33.697	74.384	151.096	251.168

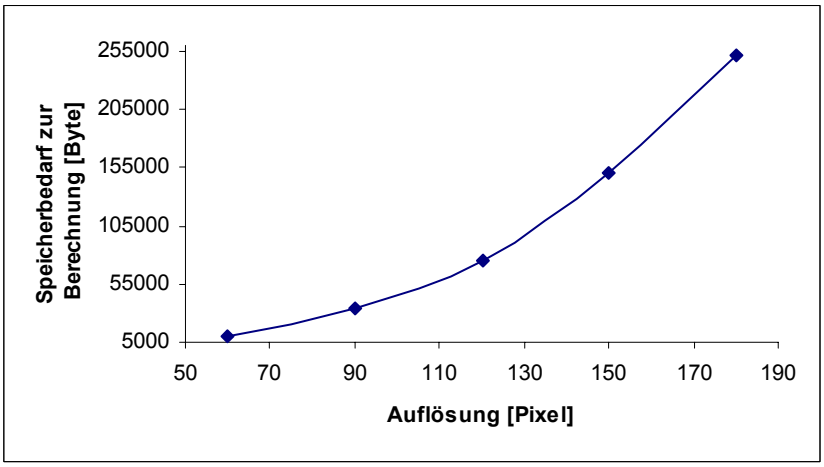


Abbildung 3-15: Speicherbedarf des Tomoalgorithmus, ermittelt mit Messwertesimulation

3.2 Spektroskopische Komponente

Eine spektroskopische Auswertung basiert im Wesentlichen auf einer Intensitätsmessung der aufgenommenen Spektrallinien. Im Plasma liegen die Gasatome in allen möglichen Anregungszuständen vor. Eine Emission von Lichtquanten findet bei jedem Rückfall eines Elektrons aus einem hohen Energiezustand in einen niedrigeren Energiezustand statt. Sie lässt sich wie folgt berechnen.

$$I_{if} = kN_i A_{if} . \tag{1}$$

Hierbei ist N_i die Anzahl der Atome im Grundzustand i, A_{if} die atomare Übergangswahrscheinlichkeit und k ein Parameter der Messanordnung, der sich mit der Wellenlänge ändert. Beispielsweise geht hier die sich verändernde Wellenlängendetektionsempfindlichkeit der Messapperatur mit ein. Es wird in Formel (1) ebenfalls die Selbstabsorbtion der Quelle vernachlässigt.
Setzt man thermodynamisches Gleichgewicht voraus lassen sich die besetzten Zustände berechnen. Thermodynamisches Gleichgewicht ist erfüllt, sobald die Dichte der freien Elektronen n_e folgende Bedingung erfüllt

$$n_e = 10^{18}\left[m^{-3}\right]\sqrt{\frac{T}{[K]}}\left(\frac{E}{[eV]}\right)^3 \tag{2}$$

E Energiedifferenz zum Grundzustand
T Temperatur

Der benötigte Zusammenhang lässt sich mit Hilfe der Boltzmann – Statistik angeben

$$\frac{n_i}{n_0} = \frac{g_i}{g_0} e^{-E_i/k_B T} . \tag{3}$$

n_i Dichte Besetzungszustand i
n_0 Dichte Grundzustand
k_B Boltzmannkonstante
E_i Energiedifferenz zum Grundzustand
T absolute Temperatur
g_i statistisches Gewicht

Summiert man alle möglichen Energiezustände können wir die Zustandssumme $Z(T)$ der Atome definieren

$$Z(T) = \sum_j g_j e^{-E_j/k_B T} . \tag{4}$$

Setzt man Gleichung (4) in Gleichung (5) ein, so erhält man die Besetzungswahrscheinlichkeit des i-ten Levels bezogen auf die totale Besetzung n

$$n_i = n \cdot \frac{g_i}{Z_Z(T)} \cdot e^{-E_i/k_B T} . \tag{5}$$

Die Nettolinienemission (Energiefluss je Volumeneinheit in den gesamten Raumwinkel) eines einzelnen Übergangs setzt sich zusammen aus der Energie eines einzelnen Lichtquants $h \cdot \nu_{if}$, der Übergangswahrscheinlichkeit A_{if} für diesen spezielle Übergang und der Teilchendichte n_i

$$\varepsilon_L = \frac{h \cdot \nu_{if}}{4\pi} \cdot A_{if} \cdot n_i . \tag{6}$$

Unter Verwendung von Gleichung (5) erhält man einen Zusammenhang zwischen Linienemission und Plasmatemperatur

$$\varepsilon_L = \frac{h \cdot \nu_{if}}{4\pi} \cdot A_{if} \cdot n \cdot \frac{g_i}{Z_Z(T)} \cdot e^{-E_i / k_B T} \tag{7}$$

bzw. mit $c = \lambda \cdot \nu$

$$\varepsilon_L = \frac{h \cdot c}{4\pi \cdot \lambda_{if}} \cdot A_{if} \cdot n \cdot \frac{g_i}{Z_Z(T)} \cdot e^{-E_i / k_B T}. \tag{8}$$

Ein Logarithmieren und Umformen führt zu folgender Gleichung

$$\ln\left(\frac{\varepsilon_L}{\left[\mathrm{Jm^{-3}sr^{-1}s^{-1}}\right]}\right) - \ln\left(\frac{A_{if} \cdot \frac{g_i}{\lambda_{if}}}{\left[\mathrm{m^{-1}s^{-1}}\right]}\right) = -\frac{1}{k_B T} \cdot E_i + \ln\left(\frac{\frac{h \cdot c}{4\pi} \cdot \frac{n}{Z_Z(T)}}{\left[\mathrm{Jm^{-2}sr^{-1}}\right]}\right). \tag{9}$$

Messwerte	Tabellierte Werte	Konstant innerhalb einer Ionisationsstufe

Trägt man den natürlichen Logarithmus der Nettolinienemission über der Anregungsenergie in ein Diagramm ein, so liegen die Werte der Übergänge innerhalb einer Ionisiationsstufe auf einer Geraden mit der Steigung $-1/k_B T$.

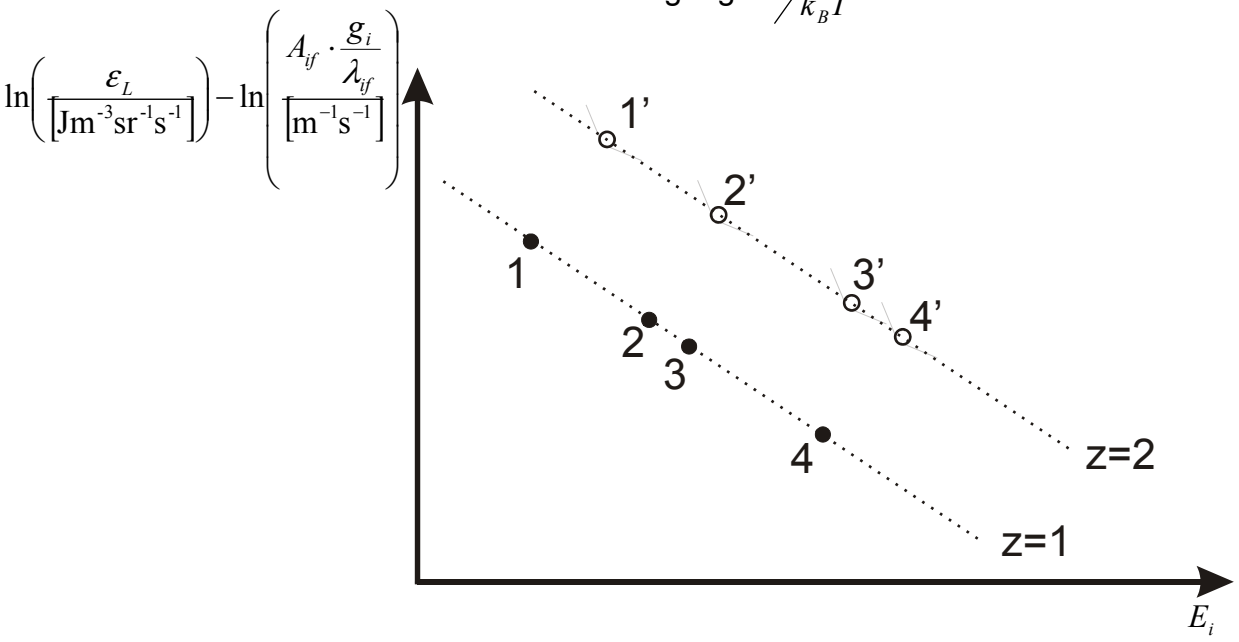


Abbildung 3-16: Prinzip Auswertung Gas- / Plasmatemperatur

Die gezeichneten Geraden stellen beispielhaft zwei Ionisationsstufen $z = 1,2$ und ihre Übergänge dar. Die in Gleichung (9) dargestellte Konstante ergibt lediglich eine vertikale Verschiebung. Die Temperatur kann aus der Beziehung

$$T = -\left(k_B \cdot gemesseneSteigung\right)^{-1} \tag{10}$$

errechnet werden.

In der Realität hängen die Besetzungsdichten der einzelnen Zustände im Plasma von vielen Prozessen ab: Kollision von Elektronen, Ionen und Neutralteilchen, Kollisionen mit Wänden, Absorption. Man muss deshalb beachten, dass die Messwerte keine reinen Linienemissionen darstellen. Es gehen alle Eigenschaften der Detektionsapparatur mit ein. Problematisch ist hier die Verbreiterung der emittierten Spektrallinien

auf Grund der Ionentemperatur. Diese Erscheinung lässt sich auf den Dopplereffekt zurückführen, der einen Zusammenhang zwischen der Frequenz des Strahlungsprozesses und der Geschwindigkeit der Strahlungsquelle herstellt, vergleichbar der Frequenzerhöhung einer sich schnell nähernden Tonquelle. Wenn sich eine Quelle mit der Geschwindigkeit u auf den Empfänger zu bewegt verändert sich die Frequenz nach folgender Beziehung

$$\nu = \nu_0 \left(1 + u/c\right), \tag{11}$$

wobei ν_0 die von der feststehenden Lichtquelle ausgestrahlte Frequenz ist, u die Geschwindigkeit der Lichtquelle und c die Lichtgeschwindigkeit. Die Wellenlänge hängt von der Frequenz ab und muss daher automatisch von u abhängen. Bei Annäherung an den Spektrografen wird sie sich verkleinern, bei der Entfernung der Lichtquelle wird sie größer.
Im Plasma bewegen sich die Atome und Ionen ungeordnet mit einer mittleren Geschwindigkeit, die zu $\sqrt{T}$ proportional ist. Die verschiedenen Teilchen werden deshalb Licht in den Spektrografen senden, das sich ein wenig von einander unterscheidet. Dies führt zu einer Verbreiterung der aufgenommenen Spektrallinien.

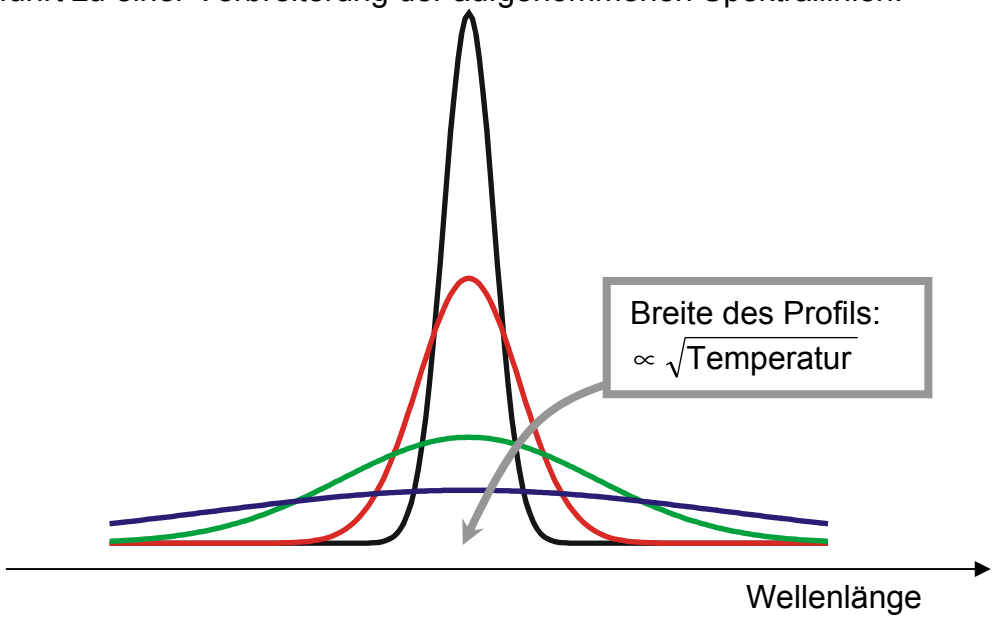


Abbildung 3-17: Veränderung des Profils der Spektrallinie

Eine Messung der Linienbreite gestattet die Bestimmung der Temperatur der Teilchen, die an dem Abstrahlungsprozess beteiligt sind.
Berücksichtigt man sämtliche Fehlerquellen (Untergrundstrahlung, Absorption etc.) und definiert hierfür einen Verbreiterungsfaktor so kann man folgende Beziehung angeben

$$\begin{aligned} \varepsilon_L &= \int_{\lambda=-\infty}^{\infty} \varepsilon_L^*(\lambda) \cdot d\lambda \\ &= \int_{\lambda=-\infty}^{\infty} \varepsilon_{L,0}^* \cdot e^{-\left(\lambda-\lambda_{L,0}/\Delta\lambda_L\right)} \cdot d\lambda \\ &= \varepsilon_{L,0} \cdot \Delta\lambda_L \cdot \sqrt{\pi} \end{aligned} \tag{12}$$

wobei es sich bei $\varepsilon_L^*(\lambda)$ um die wellenlängenspezifische Linienemission mit Störeinflüssen handelt.
Dopplerverbreiterung spielt in unserem Experiment nur eine untergeordnete Rolle. Der dominierende Faktor ist hierbei das Geräteprofil des Spektrografen. [22]

4 Hardware

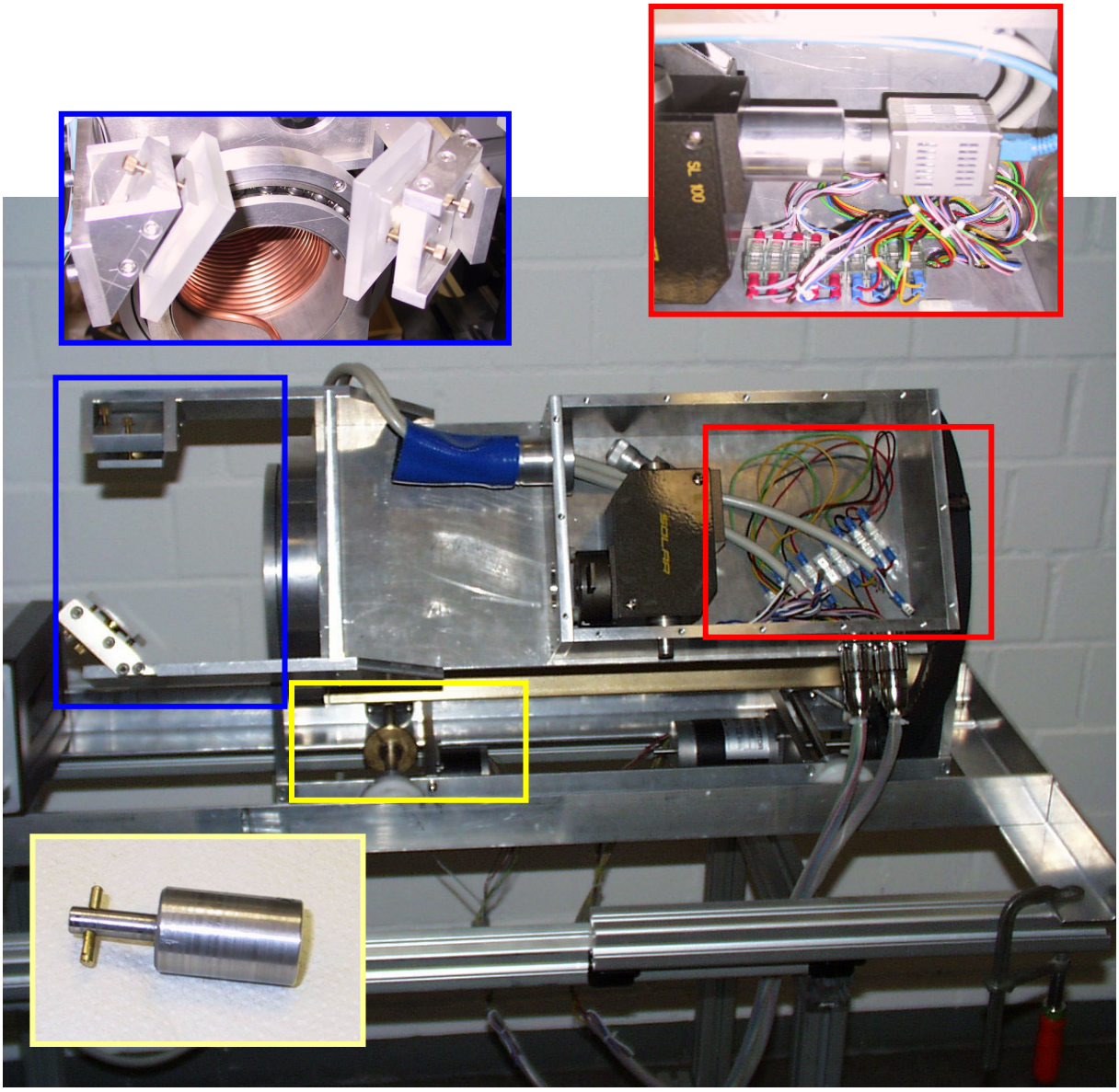
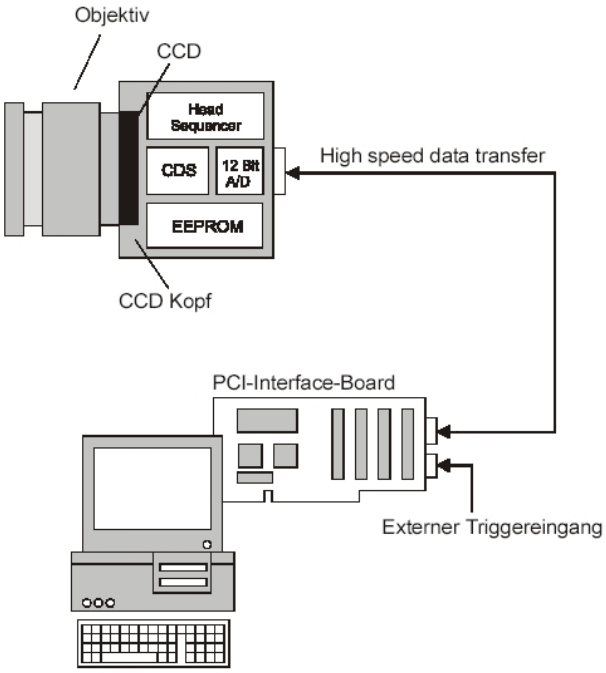


Abbildung 4-1: Gesamtaufbau

4.1 Die Kamera



Das komplette schwarz/weiß Kamerasystem besteht aus einem CCD – Kopf mit 12Bit A/D – Wandler sowie einer PCI – Interface – Karte. Unter Zuhilfenahme der mitgelieferten Software wird ein aufgenommenes Bild direkt über den PCI - Bus in den Arbeitsspeicher des Computers kopiert. Dies gewährleistet einen schnellen und effizienten Datentransfer ohne langsame Speichermedien benutzen zu müssen. Sollen die Messdaten dann als Grauwertebild am Monitor dargestellt werden, müssen diese erst von 12 Bit (4096 Graustufen) auf 8 Bit (256 Graustufen) konvertiert werden.

Abbildung 4-2: Systemstruktur

Die wichtigsten Daten der benutzten Kamera sind in folgender Tabelle zusammengestellt:

CCD Sensor	VGA
Number of Pixels	640 (H) x 480 (V)
Pixel Size	9.9 µm x 9.9 µm
Sensor Format	½“
Scan Area	6.3 x 4.8 mm
Full Well Capacity	32000 e^-
Readout Noise	16 e^-
Scan Rate	16 Mhz
Imaging Frequency …	
… at binning mode 1	40 fps
… at binning mode 2	76 fps
… at binning mode 4	140 fps
A/D Conversion Factor	7 e^-/count
Average Dark Charge @20°C (per Pixel)	< 3 e^-/sec
Spectral Range	280 … 1000 nm
Anti Blooming	> 1000
CCD Quality	Grade 0
Non – Linearity (Differential)	< 1,5%
Binning Vertical	Factor 1, 2, 4
Binning Horizontal	Factor 1, 2
Kamerakopf + PCI – Board	
Leistungsaufnahme	1 A bei 5 V, 400 mA bei 12 V
PCI – Board	Ca. 9 W
Kamerakopf VGA	Ca. 2,9 W

Der von uns verwendete VGA – Sensor hat die in Abbildung 4-3 dargestellte spektrale Empfindlichkeit.

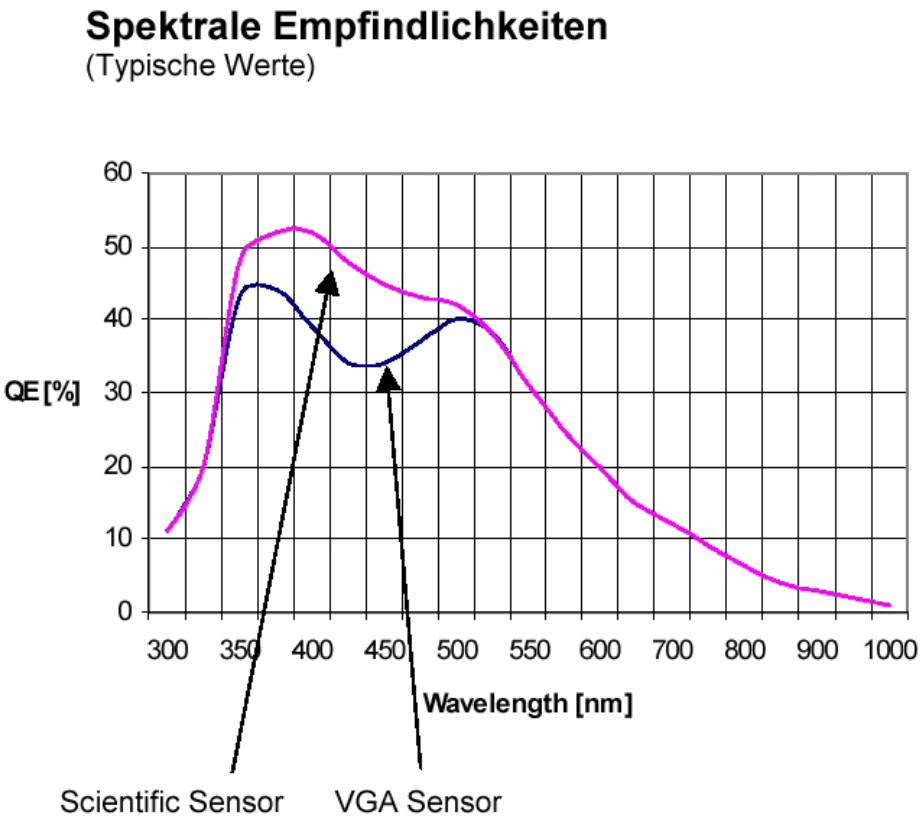


Abbildung 4-3: Spektrale Empfindlichkeit VGA Sensor (aus Datenblatt)

Die Kamerasteuerung besteht im wesentlichen aus 2 Dateien. Einer vom Hersteller mitgelieferten DLL „Pccam.dll“ sowie einer im Rahmen der Diplomarbeit entwickelten Delphi – Unit „PCO.pas“, in der die dort gespeicherten Grundlagenfunktionen zur Steuerung und Bildaufnahme benutzt werden.
Um ein Bild aufzunehmen benötigt man 3 Variablen:

```
var
        Camera: TCamera;
        pic1: TCamBuf;
        PBild: PWordArray;
```

Die Variable `Camera` vom Typ `TCamera` übernimmt die Steuerung der Kamera. Die Variable `pic1` vom Type `TCamBuf` reserviert den für ein Bild benötigten Arbeitsspeicher. Die Variable `PBild` vom Type `PWordArray` ermöglicht es dem Benutzer später direkt auf den von `pic1` reservierten Speicher zuzugreifen und beispielsweise daraus ein Bild darzustellen.
Zu erst allerdings muss der korrekte Treiber für die Kamera geladen werden (`Camera:=Tcamera.create(<BoardNr>)`). Anschließend erfolgt dann die Initialisierung (`Camera.Init()`).
Sobald diese abgeschlossen ist, kann der benötigte Speicherplatz reserviert werden (`pic1:=TCamBuf.Create(Camera.Get_Driver_Handle);`
`pic1.Allocate(<size>)`). Die Größe des benötigten Kamerapuffers hängt von der aufgenommenen Bittiefe ab. Sollen nur 8 Bit Bilder aufgenommen werden, reicht die Reservierung von 640 x 480 Byte. Bei 12 Bit Bildern, die 2 Byte Speicherplatz pro Wert benötigen, muss die doppelte Anzahl reserviert werden.

Der Befehl `Camera.GetPicture(pic1)` nimmt genau ein Bild auf und lädt dieses direkt in den zuvor reservierten Speicherbereich. Damit die Rechenzeit nicht für Busy – Waiting verbraucht wird, geschieht die Durchführung eventgesteuert. Während der Bildaufnahme und des Speicherns über den Bus des PC wird somit keine Rechenzeit verbraucht. Sollte allerdings ein Fehler auftreten, wird der Vorgang automatisch nach einer einstellbaren Timeoutzeit abgebrochen um einen Systemabsturz zu verhindern.
Nun kann durch die Zuweisung auf `PBild:=PWordArray(pic1.adr)` die Anfangsadresse ermittelt werden. Somit kann auf den von der Kamera reservierten Speicherbereich direkt zugegriffen werden. Jetzt können die Daten gespeichert oder dargestellt werden. Der Speicherbereich kann verändert werden, da die Kamera völlig unabhängig davon arbeitet. Wird allerdings ein neues Bild aufgenommen wird genau dieser Speicherbereich wieder verwendet. Das alte Bild wird somit überschrieben.
Sollen keine weiteren Bilder aufgenommen werden, muss jetzt der Speicherbereich wieder freigegeben werden. Der Befehl für den Puffer ist hier `pic1.Destroy`, für die Kamera `Camera.Free`. Die Reihenfolge muss dabei umgekehrt der Erstellung sein.

```
begin
        Camera:=TCamera.create(<BoardNr - normalerweise 0>);
                                    //Treiber laden und Board festlegen
        Camera.Init();
                                    //Kamera initialisieren + kurzer Selbstest
        pic1:= TCamBuf.Create(Camera.Get_Driver_Handle);
                                    //Buffer erstellen und mit Kamera verbinden
        pic1.Allocate(<size>);
                                    //Speicher reservieren (bei 12 bit Anzahl der
                                    Pixel mal 2 nehmen!)
        Camera.GetPicture(pic1);
                                    //Bild holen (Eventgesteuert -> kein
                                    Rechenzeitverbrauch, da anfordernder Thread in
                                    dieser Zeit inaktiv ist
        PBild:=PWordArray(pic1.adr);
                                    //Anfangsspeicheradresse einem PWordarray zuweisen
                                    -> jetzt mit z.B. PBild:=Bitmap.Scanline[<Zeile>]
                                    -> Bild ausgeben

        {... Verarbeitung der Daten ...}
        pic1.Destroy;                //Speicher freigeben Daten
        Camera.Free;                 //Speicher freigeben Kamera
end;
```

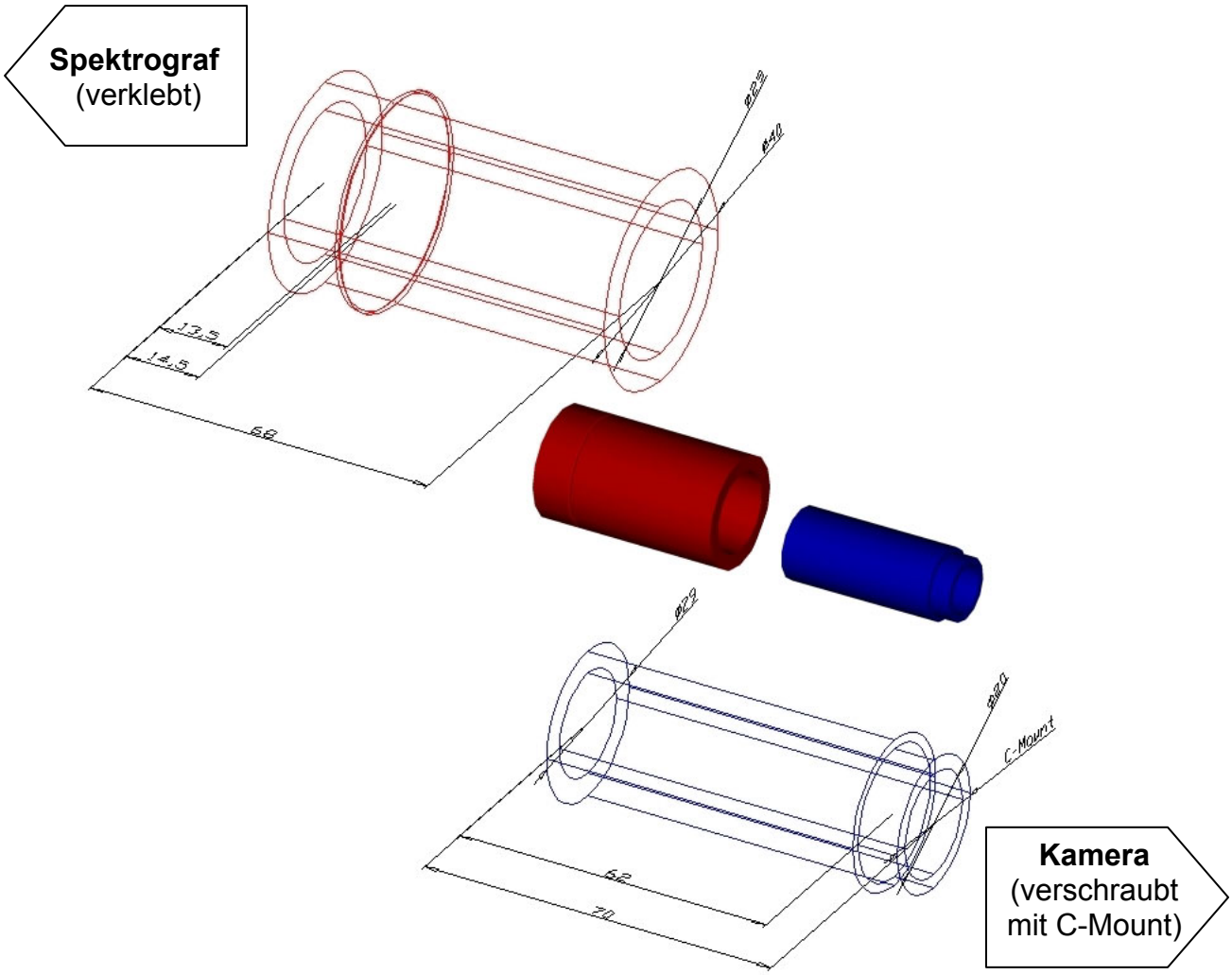


Abbildung 4-4: Aufbau Kamerahalterung

Die Kamera ist lichtdicht aber justierbar am Spektrografen befestigt. Dazu wurden zwei Rohre so gefertigt, dass sie passgenau die Kamera mit dem Spektrografen verbinden. Auf diese Weise kann kein Streulicht eindringen. Ebenfalls ist ein Verkanten der Anordnung unmöglich. Die beiden Rohre können je nach Bedarf ineinander verschoben werden. Auf diese Weise kann die Brennebene direkt auf den CCD – Kopf der Kamera eingestellt werden. Sollte die Anordnung verdreht sein, lässt sich auch das Durchdrehen der beiden Röhren ausgleichen. Um die beiden Röhren zu fixieren, können sie mit einer Feststellschraube gegeneinander blockiert werden. Die Kamera ist dabei auf die Röhre mit einem C – Mount - Gewinde aufgeschraubt. Der Spektrograf wurde mit 2 Komponentenkleber dauerhaft mit der Halterung verbunden.

4.2 Abbildungseigenschaften der Optik

An die Optik der Tomografieapparatur sind einige wesentliche Bedingungen geknüpft:

- Der Schnitt durch die Plasma-„Flamme“ muss exakt rechtwinkelig zur Drehachse der Messtrommel sein, da sonst falsche Messwerte aufgenommen werden:

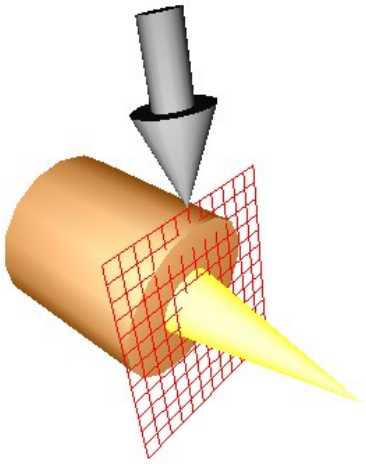


Abbildung 4-5: Fehler durch Überdecken

Fehlerquelle a: Der graue Pfeil markiert die Richtung, aus der aufgenommen wird. Da diese Richtung nicht senkrecht auf der Brennachse liegt, verdeckt der Brenner einen Teil der Flamme.

Fehlerquelle b: Der Brenner brenne in Richtung der Drehachse. Das grüne Raster symbolisiert einen Schnitt bei 0°, die rot-schwarze Fläche einen Schnitt bei 20°. Es sollte aber bei jeder Winkelposition der gleiche Schnitt durch die Flamme betrachtet werden. Hier variieren die Schnitte, was eine mit dem Verkippungswinkel zunehmende Unschärfe bewirkt, welche die Messergebnisse stark verfälscht.

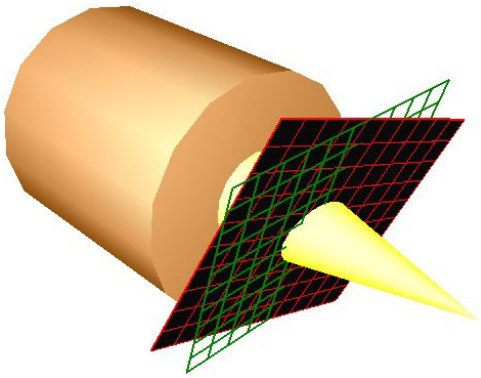


Abbildung 4-6: Unschärfe

- Die gesamte Optik darf sich infolge thermischen Einflusses auf die Messapparatur nicht verändern.
- Es dürfen keine reflektierenden Teile im Bereich der Optik sein, die Störstrahlung verursachen könnten (Schwärzen aller Halterungen und Aufbauten)
- Die eingesetzten Spiegel müssen Oberflächenspiegel sein, um dem Effekt der Doppelbilder entgegenzuwirken.

Abbildungsoptik außerhalb der Kameraschutzbox

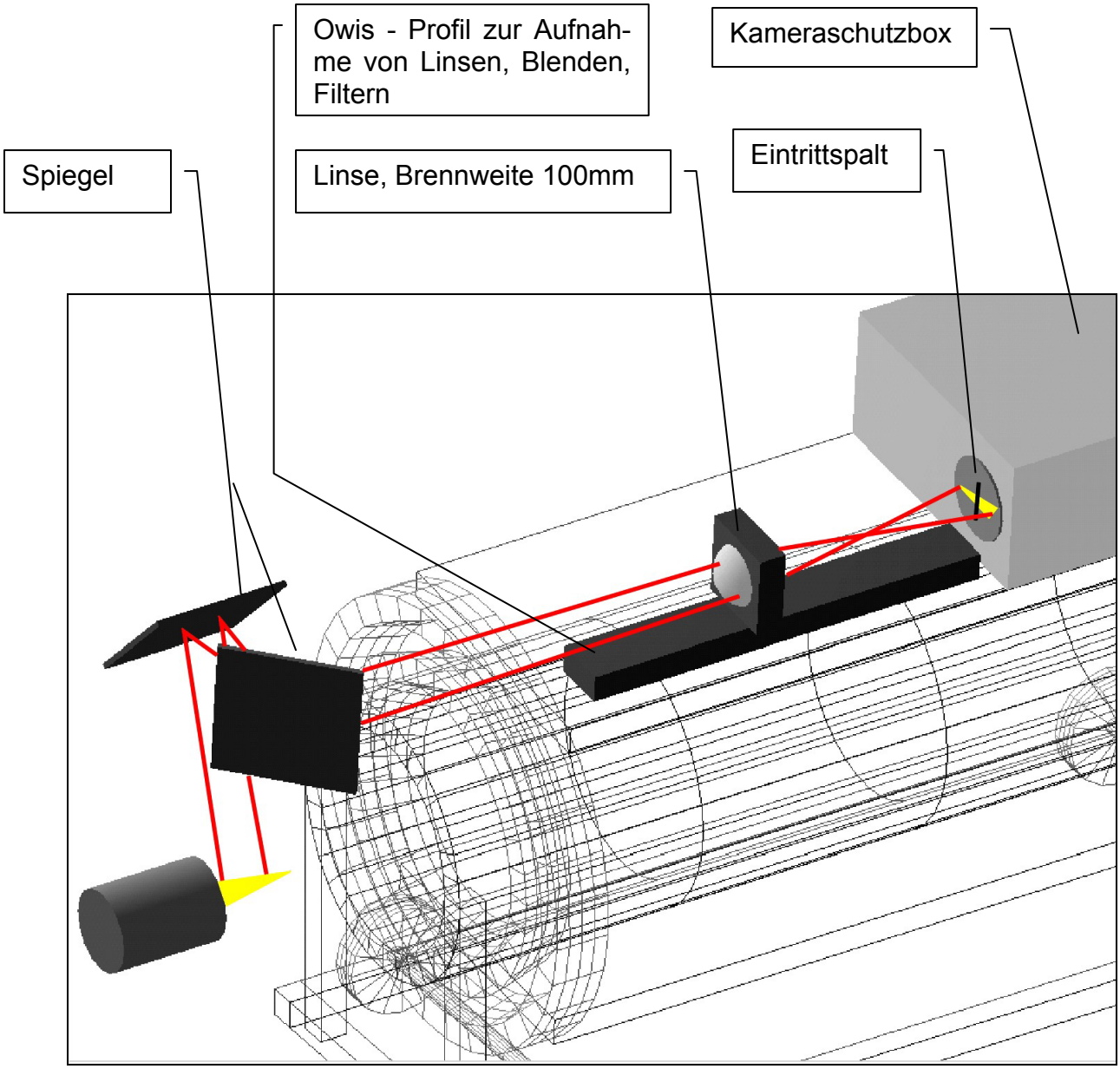


Abbildung 4-7: Strahlengang außerhalb

Die von der Plasmaflamme emittierte Strahlung wird über die beiden Spiegel durch eine konvexe Linse mit der Brennweite 100mm auf den Spalt abgebildet. Wichtig hierbei ist, wie schon oben erwähnt, dass der Schnitt durch die Flamme senkrecht zur Drehachse liegt. Am einfachsten ist dies sicherzustellen, indem das Bild des Drahtes zur Drehachseinstellung auf dem Spalt senkrecht zu liegen kommt.
Vor dem Spalt muss bei Betrieb des Plasmabrenners aufgrund der intensiven Strahlung ein optisches Filter (Verminderung der Strahlung um den Faktor 100) angebracht werden. Dieses Filter beeinträchtigt die spektralen Eigenschaften der Optik nicht; eine entsprechende Überprüfung wurde durchgeführt.

Eine andere Möglichkeit zur Verringerung der Intensität wäre eine Irisblende vor der Linse. Diese hätte jedoch eine große Tiefenschärfe der Optik zur Folge und würde die störende Wirkung jeder Verunreinigung auf den Spiegeln vervielfachen.
Abbildung 4-8 zeigt diesen Effekt.

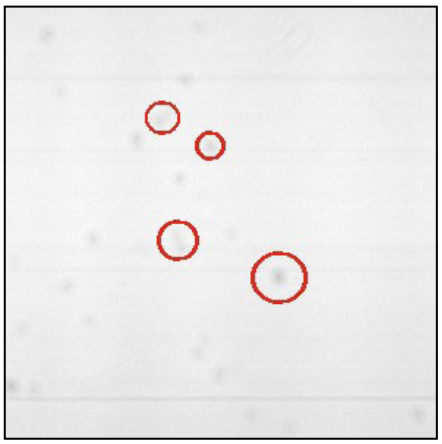


Abbildung 4-8: Verschmutzung sichtbar durch hohe Tie-

Abbildungsmaßstab

Theoretisch betrachtet beträgt der Abbildungsmaßstab gemäß folgender Grafik zwischen Objekt und dem Eintrittsspalt des Spektroprafen 1:3,85.

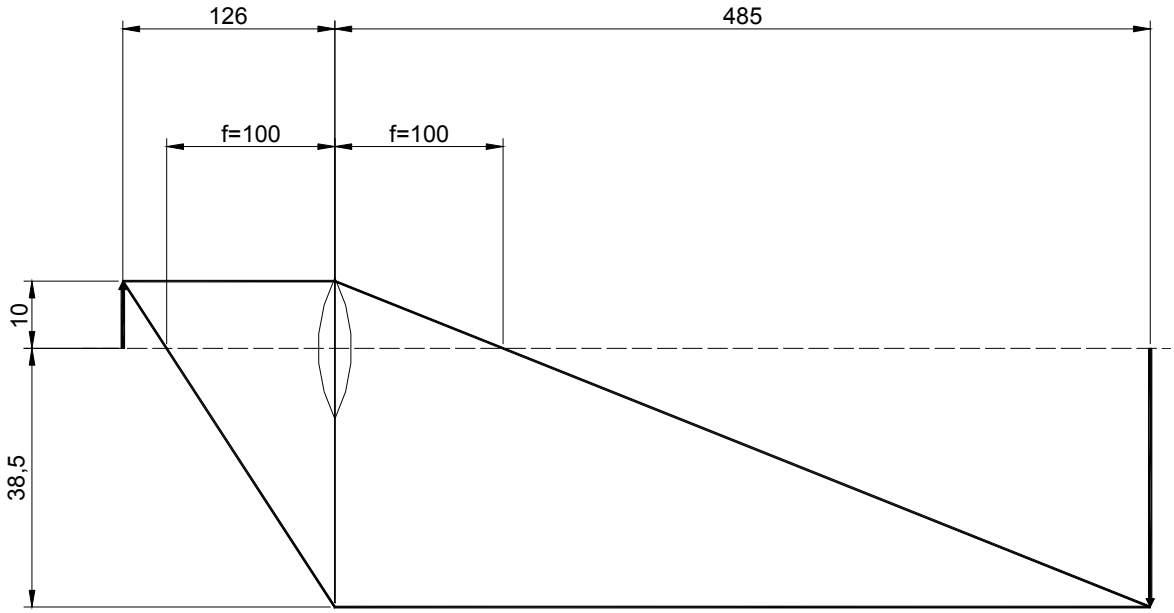


Abbildung 4-9: Grafische Bestimmung des Abbildungsmaßstabes

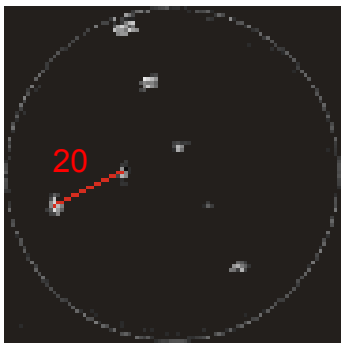


Abbildung 4-10: Experimentelle Bestimmung

Dies kann auch experimentell nachgewiesen werden, indem ein Draht in den Aufnahmebereich eingespannt und bei verschiedenen definierten Positionen aufgenommen wird. Die nebenstehende Grafik zeigt die Überlagerung von sieben aufgenommenen Bildern. Der Draht wurde jeweils um 2,5mm verschoben und dann aufgenommen. Im Bild ist diese Verschiebung um je 20 Pixel zu sehen.
Dies entspricht 60 Pixeln auf dem CCD-Chip, da zur Aufnahme drei Zeilen zusammengefasst wurden. Betrachtet man nun die Größe des CCD-Chips der Kamera von 6,3*4,75mm bei einer Auflösung von 640*480Pixel (Pixelgröße 9,9*9,9µm^2), so lässt sich der Abbildungsmaßstab von 1:4,2 berechnen. Hier wäre noch anzumerken, dass

der Abbildungsmaßstab des Spektrograf nicht ganz 1:1beträgt. Daher ergibt sich die minimale Abweichung zur grafischen Bestimmung.

Berechnung der minimal möglichen Größe der Spiegel

Im Rahmen von Verbesserungsmaßnahmen werden die vorhandenen Spiegelfließen durch Oberflächenspiegel ersetzt: Sie sind zum Einen zu groß dimensioniert, sodass sie bei Messungen mit dem Messobjekt in Berührung kommen könnten. Zum Anderen soll „Doppelbildern“ mit Hilfe der Oberflächenspiegel entgegengewirkt werden.
Die einfachste Variante, die minimal mögliche Spiegelgröße zu bestimmen ist die grafische Darstellung des Strahlengangs:

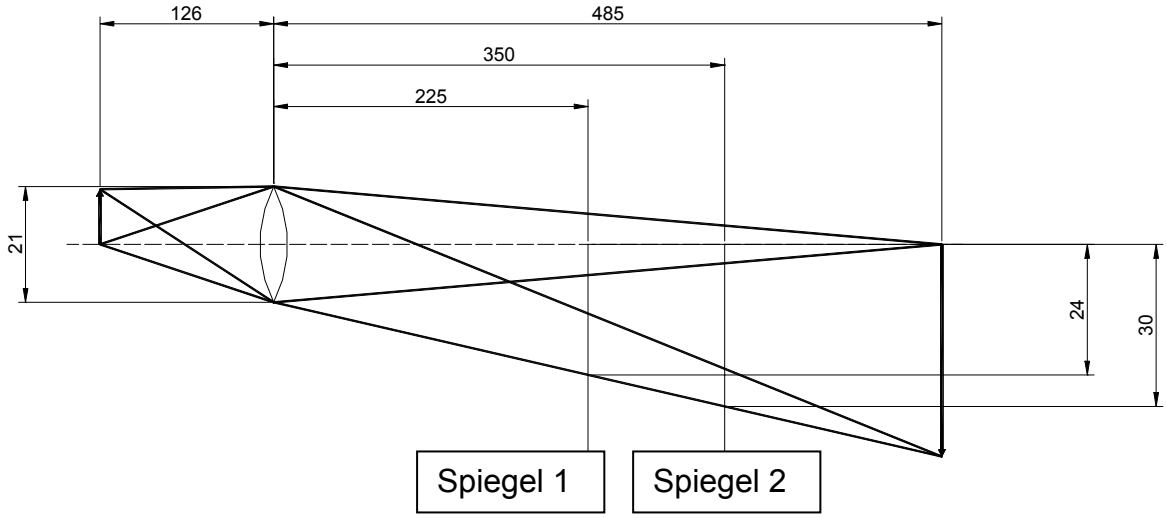


Abbildung 4-11: Ermittlung der minimal möglichen Spiegelgröße

Zusätzlich muss man betrachten, dass die wirksame Spiegelfläche nicht der Breite der Spiegel entspricht. Am ungünstigsten wirkt sich dies auf Spiegel eins aus, der um 45 Grad gedreht zur optischen Achse der Linse angebracht ist. Dieser muss demnach mindestens $24mm \cdot 2 \cdot \sqrt{2} \approx 68mm$ Kantenlänge haben.
Die tatsächlich eingesetzten Spiegel sind quadratisch mit einer Kantenlänge von 80mm. Dies reicht für Spiegel zwei und, nach Rechnung, für Spiegel eins völlig aus, um den kompletten Raumwinkel abzubilden.

Abbildungsoptik innerhalb der Kameraschutzbox

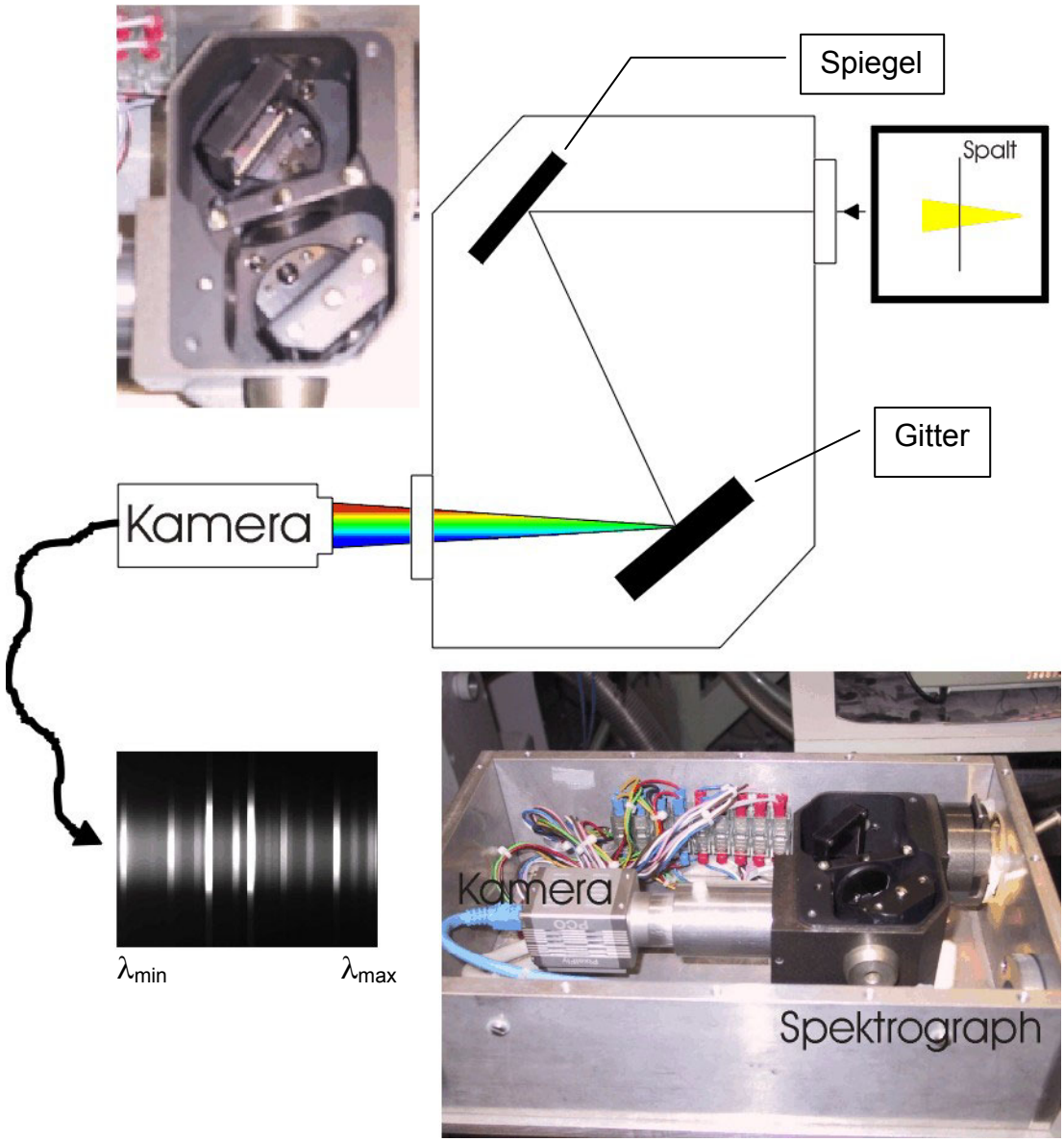


Abbildung 4-12: Strahlengang innerhalb der Kameraschutzbox

Das von dem Spiegel reflektierte Licht trifft auf ein Gitterspektrometer, wo es in seine Spektralanteile zerlegt wird. Das Gitter ist drehbar gelagert, so dass der spektrale Bereich zwischen λ=560nm und λ=790nm variiert werden kann.

In Abbildung 4-13 ist dargestellt, welcher Wert an der Mikrometerschraube des Spektrografen einzustellen ist, um mit der Messapparatur einen spektralen Bereich begrenzt durch linke und rechte Wellenlänge zu betrachten:

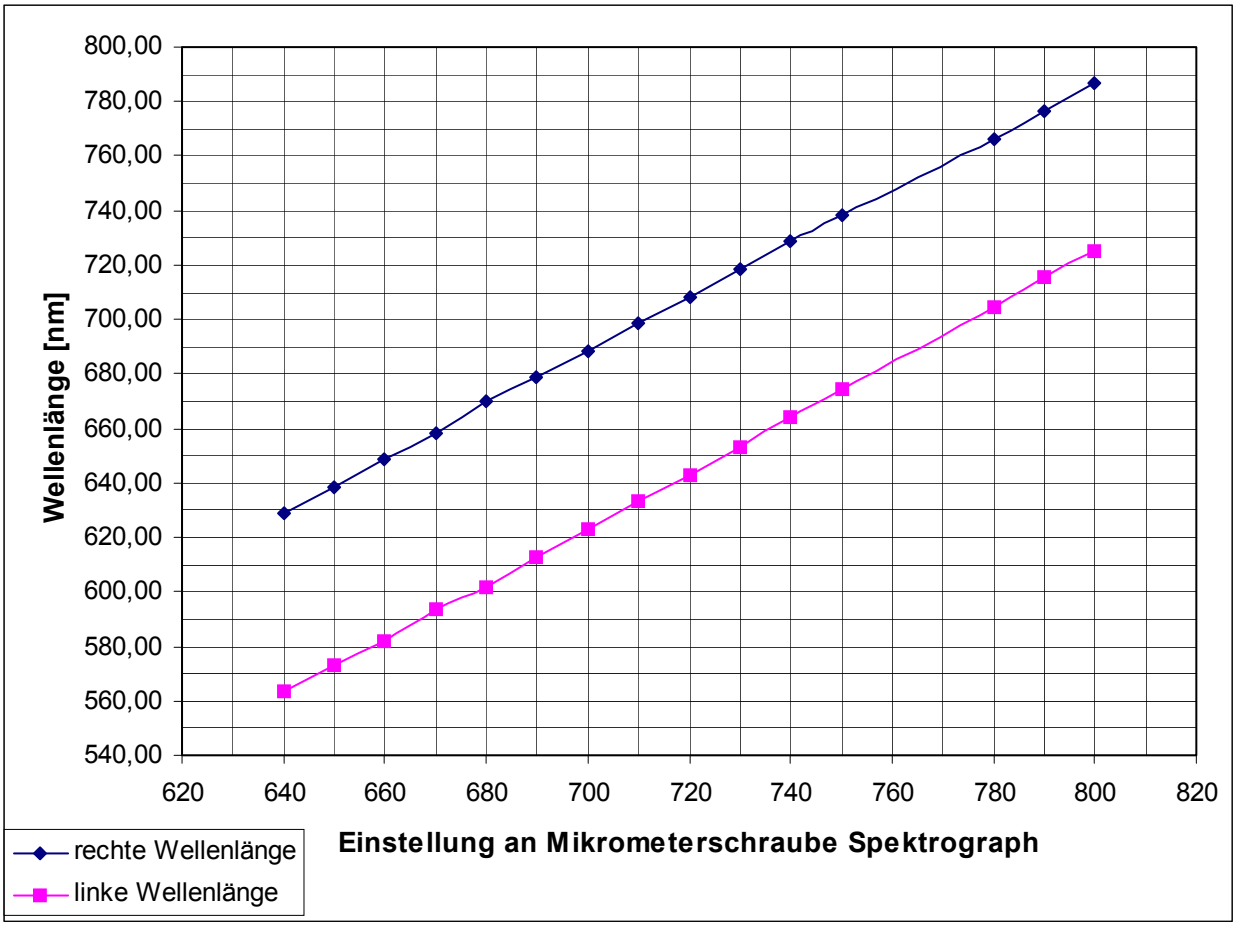


Abbildung 4-13: Einstellungsmöglichkeiten am Spektrografen

Anmerkung: Dieses Diagramm wurde mit Hilfe der Spektral-Justage-Einrichtung erstellt! (siehe „Das Formular Spektraljustage“)

Justage der Optik

Wie schon oben erwähnt, muss die Apparatur „senkrecht auf die Drehachse blicken“. Als die praktikabelste Lösung zur Justage ergab sich die Folgende:

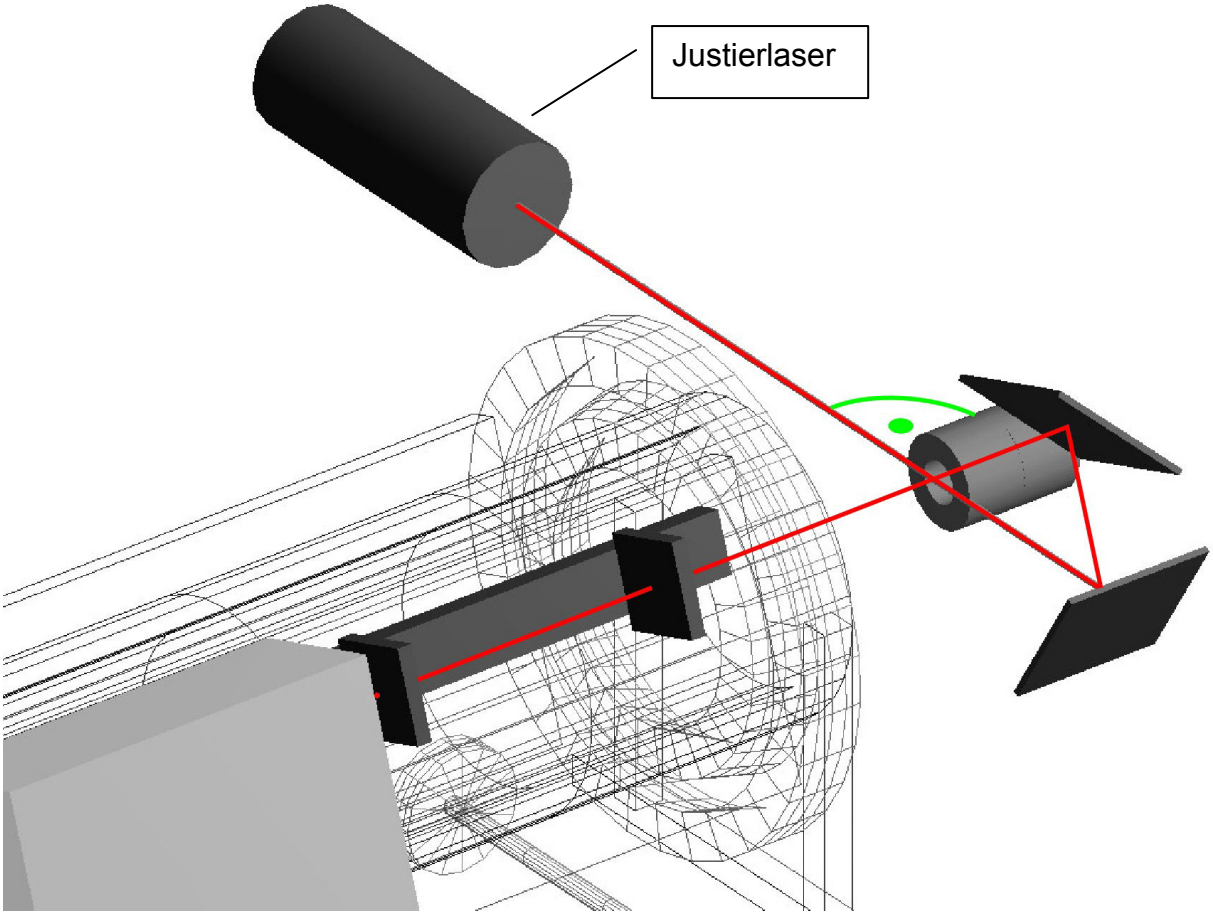


Abbildung 4-14: Justage der Optik

Ein Justierlaser wird exakt senkrecht zur Drehachse an der Halterung der Messapparatur befestigt (siehe Abbildung 4-14). Nun müssen die Spiegel so eingestellt werden, dass das Laserlicht durch die beiden Lochblenden auf den Eintrittsspalt fällt.

4.3 Die Wellenkupplung

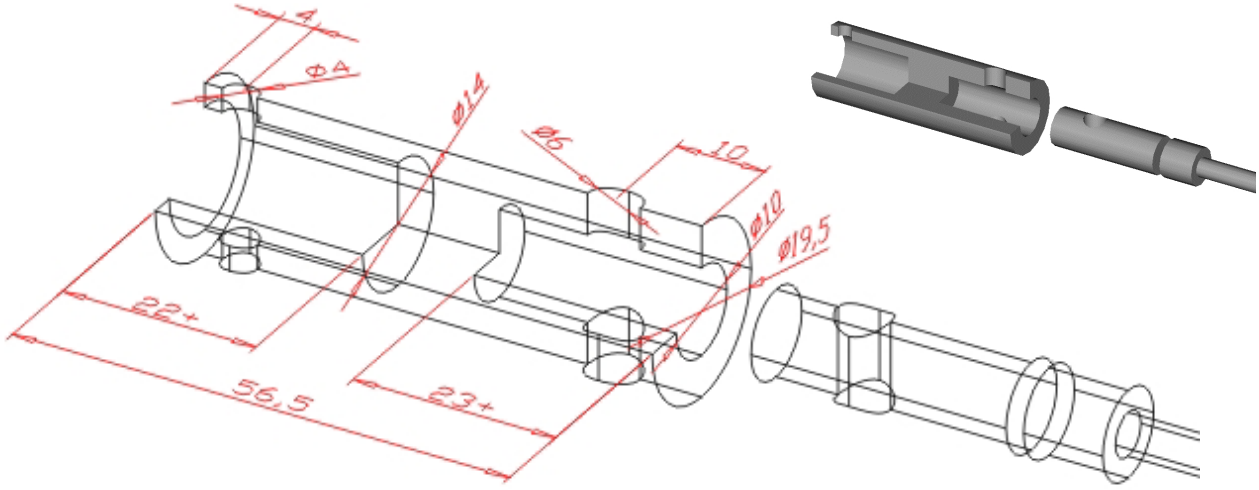
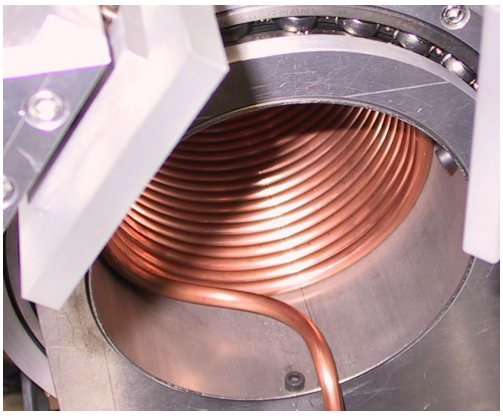


Abbildung 4-15: Wellenkupplung

Nach Erfahrungen im Lauf der Programmerstellung und Versuchsdurchführung wurde die vorhandene starre Welle durch eine Wellenkupplung ersetzt. In diese wird ein Stift eingesetzt, der bei Blockierung der Trommel abgeschert wird. Auf diese Weise wird eine Beschädigung des Getriebes und anderer Teile der Welle verhindert. Der Austausch dieses Stiftes gestaltet sich einfach, da dieser leicht zugänglich ist. Eine Beschädigung an einer anderen Stelle hätte sonst stets einen langen Ausfall der Tomografieeinrichtung zur Folge.

4.4 Die Kühlschlange



Um den Dauerbetrieb des Messaufbaus zu ermöglichen musste eine 35 m lange Kühlschlange aus Kupfer eingebaut werden. Auf diese Weise ist es möglich die Messapparatur auch bei langen Messreihen auf konstant niedriger Temperatur zu halten.

Abbildung 4-16: Kühlschlange

5 Software

Programmierumgebung und Systemanforderung:

Als Programmierumgebung wurde Delphi 6.0 Enterprise gewählt. Es handelt sich hierbei um eine moderne objekt- und ereignisorientierte Programmiersprache. Um ein effektives Arbeiten zu ermöglichen werden verschiedene Editoren mitgeliefert.
Neben den mitgelieferten Paketen mussten mehrere externe Komponenten installiert werden, ohne die die entwickelte Software nicht lauffähig ist. Dies sind:

- Serieller Schnittstellentreiber[1] zum Betrieb der Schrittmotorsteuerung
- OpenGl – Editor – Sammlung[2] zur besseren visuellen Darstellung des Betriebs und der ausgewerteten Messdaten
- Windowslaufzeitbibliothek zur Steuerung der Kamera (Pccam.dll)

Mindestsystemvorrausetzungen laut Hersteller:

- Intel Pentium 233 MHz
- Microsoft Windows XP, Windows 2000 oder Windows 98
- 64 MB Ram
- 124 MB HDD
- CD – Rom Laufwerk
- SVGA – Grafikkarte

Das entwickelte Programm dient zur tomografischen Auswertung aufgenommener Messdaten. Um ein effektives Arbeiten zu ermöglichen, sollte das System deutlich leistungsfähiger sein. Gute Ergebnisse wurden auf folgenden Systemen erzielt:

- Intel Pentium IV 2 GHz bzw. AMD Athlon XP 1700+
- Microsoft Windows XP, Windows 2000
- 512 MB Ram
- min. 500MB HDD (ohne Messdaten)
- SVGA – Grafikkarte mit einer Auflösung von 1280x1024

Kritisch hierbei sind vor allem Arbeitsspeicher, Festplattenplatz und Auflösung der Grafikkarte. Das Projekt wurde für die angegebene Auflösung entwickelt. Es wurde sichergestellt, dass kein Formular größer als der am Bildschirm angezeigte Bereich ist. Das Programm beinhaltet die Möglichkeit alle aufgenommenen Bilddaten zur späteren Auswertung zu speichern. Es entstehen dabei Messreihen von bis zu 108 MB. Genügend Festplattenplatz sollte deshalb sichergestellt werden. Rechner mit Arbeitsspeicher von weniger als 512 MB sollten nicht verwendet werden, der Tomografiealgorithmus ist sehr speicheraufwendig. Bei weniger Arbeitsspeicher muss auf HDD ausgelagert werden. Auf Grund der hohen Zugriffszeiten führt dies zu einer extremen Rechenzeitzunahme.

[1] Marco Cocco: TCommPortDriver component, COM Port Driver for Delphi 2.0, v1.08/32 - November 19th, 1997.

[2] Eric Grange: OpenGl – Library, Copyright 2000-2002. (www.GlScene.org)

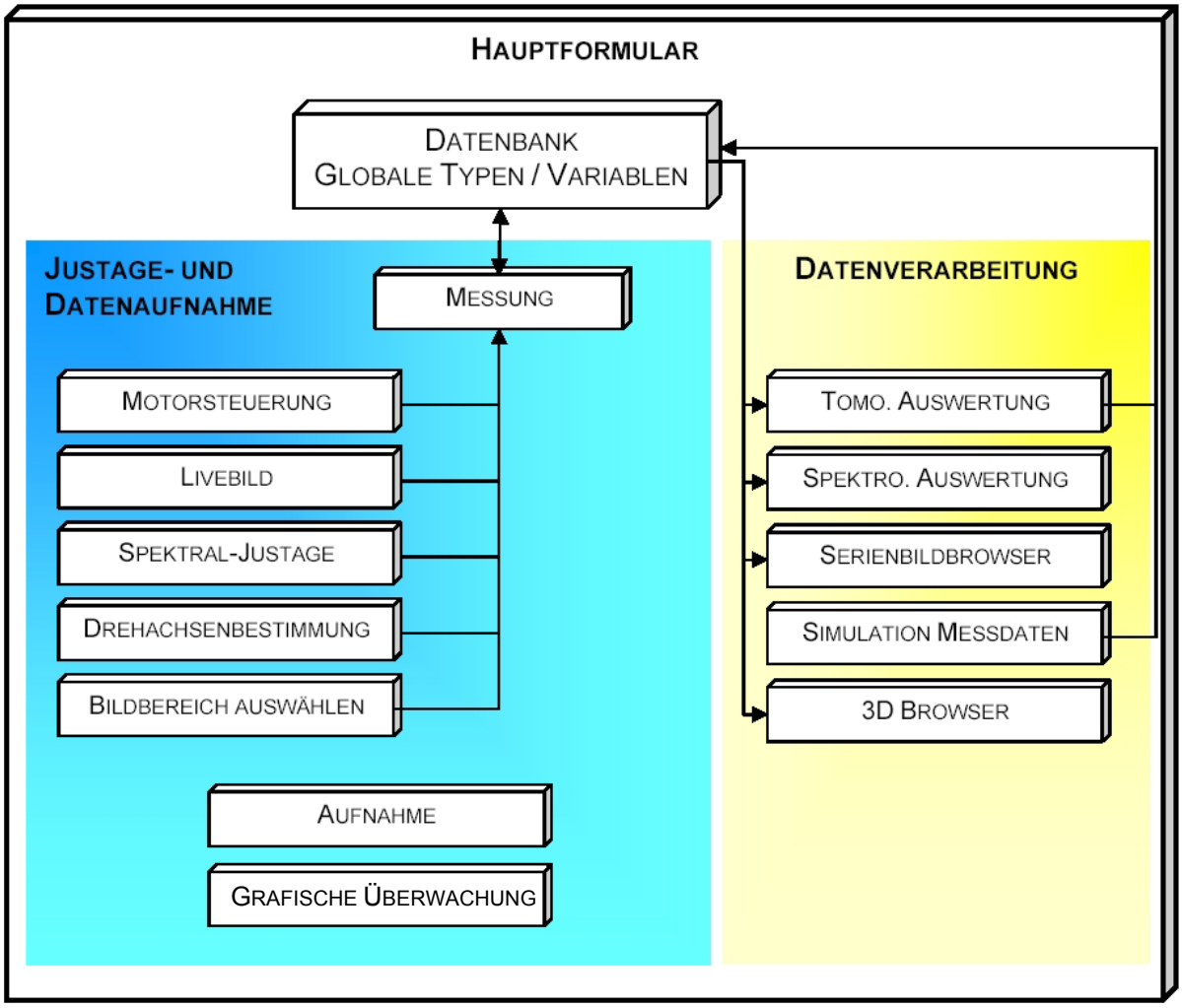


Abbildung 5-1: Zusammenspiel Funktionseinheiten

Abbildung 5-1 zeigt das Zusammenspiel der beiden großen Funktionsgruppen sowie ihre Verbindung zur Datenbank. Die Pfeile zeigen die Richtung des Datenstromes an. Aus Gründen der Übersichtlichkeit wurde in der Darstellung auf Hilfsformulare verzichtet. In den anschließenden Kapiteln werden nun alle Formulare und ihre Aufgaben beschrieben.

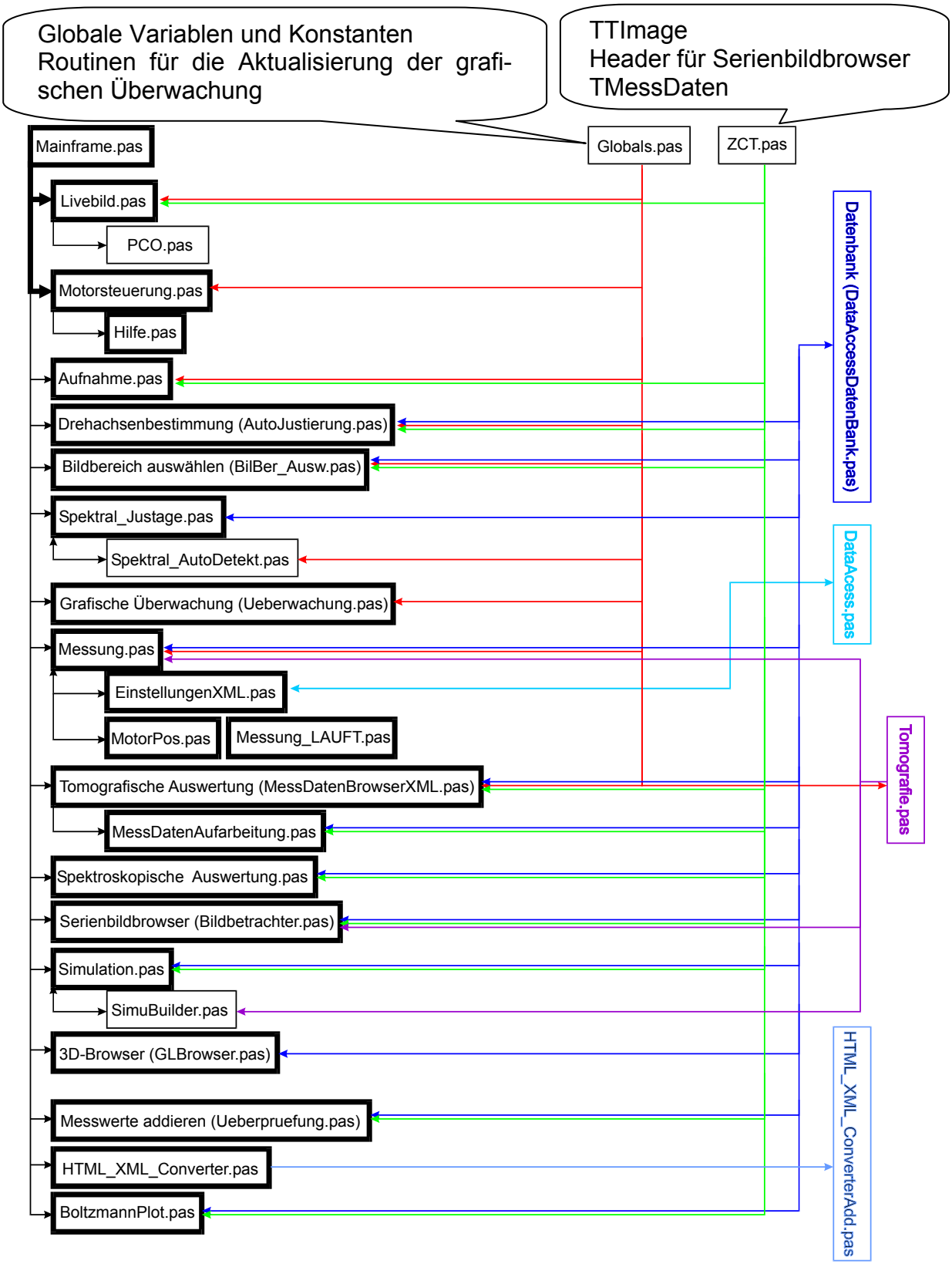


Abbildung 5-2: Übersicht über die Zusammenhänge zwischen den Projektdateien

Erläuterung:

Fette Umrandung: Die Unit hat eine eigene Form.

Dünne Umrandung: Die ist lediglich ein Hilfskonstrukt und hat keine eigene Form.

Ein fetter schwarzer Pfeil bedeutet, dass die Form bei Programmstart geladen wird.

5.1 Das Hauptformular

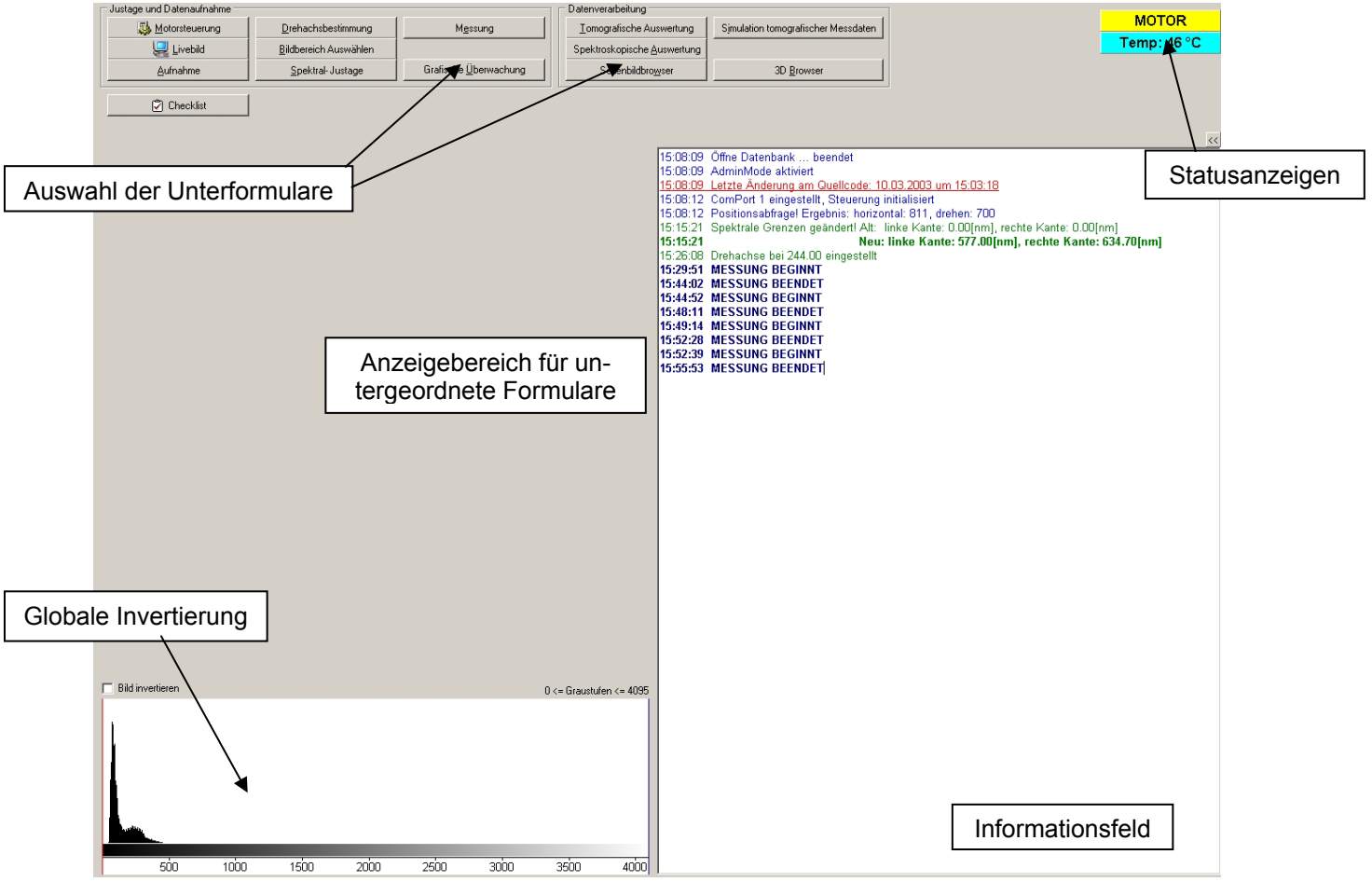


Abbildung 5-3: Das Hauptformular

Das Hauptformular hat die Aufgabe, Statusinformationen über die beiden externen Komponenten der Steuerung (Motor und Kamera) zur Verfügung zu stellen und dient als zentrale Steuereinheit. Von hier können alle Unterformulare gestartet und geschlossen werden. Es bietet zusätzlich in der unteren rechten Ecke ein Informationsfeld, in dem während der Programmausführung wichtige Meldungen ausgegeben werden können. Um Testmessungen durchführen zu können, gibt es eine Möglichkeit durch globale Invertierung der aufgenommenen Bilder, auch nichtemittierende Messobjekte (wie z.B. Stecknadeln) aufzunehmen, um die räumliche Auflösung der Messanordnung zu verifizieren. Ebenfalls verhindert es einen Mehrfachstart des Programms.
Um den Zustand der angeschlossenen Komponenten anzuzeigen befinden sich in der oberen rechten Ecke zwei Statuspanels. Das erste stellt den Status der Motorsteuerung dar. Es hat drei Farben mit folgenden Bedeutungen:

- Grün: Motorsteuerung ist noch nicht initialisiert
- Gelb: Programm hat die Kontrolle über die Motorsteuerung ist jedoch in StandBy
- Rot: Programm sendet Befehle an die Motorsteuerung und bewegt die Messanordnung

Auf diese Weise ist sofort ersichtlich welche Gefahr vom Messaufbau ausgeht. Sobald der Computer die Kontrolle übernommen hat (Anzeige auf gelb) darf aus Sicherheitsgründen nicht am Messaufbau gearbeitet werden.

Bei der Kamera ist die Kodierung anders, da diese keine Gefahrenquelle darstellt:

- Weiß: Kamera nicht funktionsfähig (nicht angeschlossen / defekt)
- Rot: Kamera nicht funktionsfähig (muss erst gestartet werden)
- Gelb: Kamera initialisiert
- Blau: Kamera funktionsfähig nimmt allerdings keine Bilder auf
- Grün: Kamera zeichnet Bilder auf

Im Abstand von fünf Sekunden zeigt dieses Panel die Temperatur der Kamera an. Die Betriebstemperatur beträgt ca. 50 °C. Im Bereich von –55 °C bis +125 °C ist die Kamera funktionsfähig. Im vorgegebenen Einsatzfeld ist eine Abkühlung der Kamera ausgeschlossen. Deshalb wird vom Programm ab 100 °C eine Warnmeldung ausgegeben. Spätestens dann sollte der Betrieb eingestellt werden und die Kamera wieder abgekühlt werden.
Rechts unten auf dem Hauptformular befindet sich ein Meldungsfenster. In diesem können von jedem Formular Nachrichten farbkodiert eingetragen werden. Dies geschieht zum Beispiel während der Messung oder wenn globale Werte ermittelt wurden.
Links unten befindet sich eine Groupbox, in der eine Invertierung des aufgenommenen Bildes vorgenommen werden kann. Auf diese Weise kann auch ein Nichtstrahlendes Objekt tomografiert werden. Diese Messungen sind sehr gut geeignet zur Ermittlung der Genauigkeit der Messanordnung. Nimmt man einen Draht, der die Breite genau eines Pixels hat, an verschiedenen Positionen auf und vergleicht dann die rekonstruierten Positionen mit den realen, so lässt dies Rückschlüsse auf die Genauigkeit der Messanordnung zu (s. Kap. 5.4.4).
Zur Auswahl der Unterformulare stehen am oberen Rand des Formulars mehrere Schaltflächen zur Verfügung. Die Formulare sind bei Programmaufruf noch nicht erstellt. Sie werden durch Druck auf die Schaltfläche erzeugt und angezeigt und bei erneutem Klick wieder freigegeben. Da die Formulare geschlossen werden, bleibt die Information nicht erhalten und das Formular wird durch erneutes Anzeigen verändert. Deshalb speichern alle Formulare wichtige Variablen global, um sie bei erneutem Start wieder aufrufen zu können. Eine Ausnahme hier bilden die beiden Formulare Motorsteuerung und Livebild. Diese beiden Formulare werden einmal erstellt und bleiben dann erhalten. Dies ist hier sinnvoll, da es sich um zwei Kommunikationsformulare handelt. Sie stellen die Verbindung zur Kamera und zur Schrittmotorsteuerung her und halten diese während des gesamten Programmverlaufs aufrecht.
Die jeweiligen Formulare bleiben stets im Vordergrund der Anwendung, denn sie haben eine höhere Anzeigepriorität als das Hauptformular.

Eine der wichtigsten Funktionen des Hauptformulars ist es, einen Mehrfachstart zu verhindern. Dafür sind die folgenden Zeilen notwendig:

```
mHandle: THandle;

Initialization   //Mehrfachstart verhindern
  mHandle:=CreateMutex(nil,True,'AKINA');
  if GetLastError=ERROR_ALREADY_EXISTS then begin
    MessageBox(0, 'Eine Instanz dieses Programms läuft bereits.' + #13 + 'Programmstart
               abgebrochen!', 'AKINA', mb_IconHand);
    Halt;
  end;

finalization
  if mHandle<>0 then
    CloseHandle(mHandle)
```

Bei jedem Start des Programms wird ein Handle mit dem Titel „AKINA“ erzeugt. Um seine Eindeutigkeit zu erhalten darf dieser Handle jeweils nur einmal vorkommen. Sollte der Handle mehrmals vorkommen wird von Delphi die Fehlermeldung

„ERROR_ALREADY_EXISTS ausgelöst. Diese wird mit GetLastError abgefragt. Sollte diese Fehlermeldung ausgelöst worden sein, wird der Benutzer durch eine Nachricht auf dem Bildschirm darauf hingewiesen. Anschließend wird das Programm mit Halt abgebrochen. Im finalization Teil wird der Handle dann wieder geschlossen. Der zweite Programmstart wurde somit komplett abgebrochen.
Die Darstellung des Informationsfensters wurde als RichEdit realisiert. In diesem ist es ohne Probleme möglich, farbkodiert zu schreiben. Dafür ist die public Procedure NOTIFY zuständig:

```
procedure TMainframe_Form.NOTIFY(Text: String; Const Color: integer = clBlack; Const Stil:
TFontStyles =[]; Const DeleteLast: Boolean = False);
begin
        if DeleteLast then Info.Lines.Delete(Info.Lines.Count-1);
        Info.SelAttributes.Style:=Stil;
        Info.SelAttributes.Color:=Color;
        Info.Lines.Add(TimeToStr(Time)+'  '+ Text);
end;
```

Diese Procedure wird mit mehreren Parametern aufgerufen:

- Text: Anzeigetext im Feld
- Color: Farbe des Textes (Voreinstellung schwarz)
- Stil: Formatierung des Textes wie Fett oder Kursiv (Voreinstellung keine)
- DeleteLast: Löschen der letzten angezeigten Zeile (Voreinstellung nicht löschen)

Nachdem die Attribute gesetzt wurden, wird der Text mit aktueller Systemzeitangabe in das Informationsfeld geschrieben.
Um unterscheiden zu können welches Formular invertiert dargestellt werden muss, steht die Procedure FormNeuMalen zur Verfügung:

```
TMaleFunction = procedure of Object;
TMale = Record
                Target: TMaleFunction;
                Sender: ^Boolean;
                end;
malArray: array of TMale;

procedure TMainframe_Form.FormNeuMalen();
var
        i: Integer;
begin
    for i:=0 to Length(MalArray)-1 do begin
        if MalArray[i].Sender^ then MalArray[i].Target;
    end;
end;
```

Jedes Fenster, das die Möglichkeit hat sinnvoll invertierte Daten darzustellen, kann sich bei seiner Erstellung in den Array malArray eintragen. Ist nun die Invertierung aktiviert, wird geprüft welches Fenster neu gezeichnet werden muss. Dies geschieht durch Auswertung der booleschen Variable Sender. Dies ist ein Zeiger auf einen vom Formular selbst verwalteten booleschen Ausdruck, in der Regel auf die Eigenschaft Visible des Formulars. In Target wird die Funktion gespeichert, die für das Neuzeichnen des Formulars zuständig ist. Nun kann durch Abfrage der Schleife in FormNeuMalen entschieden werden, ob das Fenster neu gezeichnet werden soll. Würde man alle Fenster neu zeichnen, wäre dies zu zeitaufwendig und die Anwendung hätte Probleme auf Grund zu geringer Rechenzeit.
Das Hauptformular öffnet ebenfalls bei seiner Erstellung die Datenbankdatei der Anwendung. Sollte dabei ein Fehler auftreten, wird dieser sofort im Informationsfeld angezeigt. Eine weitere Programmausführung ist dann nur bedingt möglich, da keine Daten gespeichert werden können. Es besteht allerdings die Möglichkeit eine leere Datenbank zu erstellen. Wird die leere Datenbank nicht erstellt, kann das Programm nicht richtig verwendet werden.

```
    try
          XMLDocumentDataBase:=TXMLDocument.Create(ExtractFilePath(Application.ExeName) +
               'MessDaten\DatenBank.XML');
          DataBase:=GetDatenbank(XMLDocumentDataBase);
          Notify('Öffne Datenbank ... beendet', clBlue, [], TRUE);
    except
          Notify('KRITISCHER FEHLER --> Datenbank kann nicht geöffnet werden ...
                 PROGRAMMFUNKTION ohne DATENBANK NICHT MÖGLICH', clRed, [fsBold], FALSE);
          if Application.MessageBox('Leere Datenbank erstellen?', 'AKINA'
                                MB_YESNO)=IDYES then begin
               XMLDocumentDataBase:=TXMLDocument.Create(Mainframe_Form);
               XMLDocumentDataBase.GetDocBinding('Datenbank', TXMLDatenbankType);
               XMLDocumentDataBase.SaveToFile(ExtractFilePath(Application.ExeName) +
                   'MessDaten\Datenbank.xml');

               XMLDocumentDataBase:=TXMLDocument.Create(ExtractFilePath(Application.ExeName)
                    + 'MessDaten\DatenBank.XML');
               DataBase:=GetDatenbank(XMLDocumentDataBase);
               Notify('Leere Datenbank erstellt ...', clBlue, [], FALSE);
          end;
    end;
```

5.2 Globale Typen, Variablen und Konstanten

5.2.1 Variablen

Name	Typ	Beschreibung
LiveBild		
GetScreenShot	Boolean	Falls TRUE, Livebild wird in den Speicher geladen
DrawScreenShot	TMaleFunction	Procedure des Objektes, welches von LiveBild aufgerufen wird nach erfolgreich abgelegtem Bild
TomoAlgorithmus		
ABRUCH_FLAECHEN FAKTOREN_BERECHNUNG	Boolean	Falls TRUE, Berechnung der Flächenfaktoren wird abgebrochen
AUFLOESUNG_GEAENDERT	Boolean	TRUE, falls Auflösung geändert
SIMULATION	Integer	Speicher für Auflösungswert
Aufloesung	Integer	
Iterations_Schleifen	TList	Listen für Tomografiealgorithmus
Bild_Orginal	TList	
Bild_Differenz	TList	
Liste	TList	
Messwerte_Simuliert	TList	
Istwerte_alt	TList	
Istwerte	TList	
Messwerte_Buffer	TList	
Messwerte	TList	
Sum_FF	TList	
Messwerte_Ist	TList	
Gew_Feld	TList	
Max_Diff_Feld	TList	
Kamerabild		
BildDaten12bit	^TBild	Zeiger auf Bildspeicherbereich
EVENTS		
FahrEvent	TEvent	Event für Motorsteuerung
RefeEvent	TEvent	Event für Referenzfahrt
JustageEvent	TEvent	Event für Justage
BildEvent	TEvent	Event für Bild eingegangen

5.2.2 Konstanten

Name	Beschreibung
Spektral_AutoDetekt	
Detektionsebene=0.22	
Spektral_Justage	
links_Vorgabe = 572.5	Startwerte für Wellenlängenachse [nm]
rechts_Vorgabe = 661.1	
Spektrallinienbrei- te_Vorgabe = 13	
Aufnahme	
AuswerteBildHeight=480	Bilddaten
AuswerteBildWidth=640	
AuswerteBildHistoHeight=256	
AuswerteBildHistoWidth=256	
Farbtiefe=256	
TomoAlgo	
Max_Aufloesung = 180	
Motorsteuerung	
Initialisierung='@03'	Motorbefehle
Positionsabfrage='@0P'	
Nothalt=CHR(255)	
XReferenzfahrt='@0R1'	
YReferenzfahrt='@0R2'	
XMax = 96000	Anzahl Schritte bis Endposition (fahren)
YMax = 92840	Anzahl Schritte bis Endposition (drehen)
XMaxcm = 43	Entfernung Anfangs- zu Endposition in cm
YMaxgrad = 184	Winkel zw. Anfangs- zu Endposition in grad
Nullpunkt = 1{[°]}	Einstellung des Nullpunktes in Grad
TBarFaktor = 10	
ActionPosX = 10000	Für Autojustierung
StartAngle = 700	
WorkAngle = 91913	
EinstX = 63440	Drehposition zur Einstellung X-Achse
EinstY = 15000	Drehposition zur Einstellung Y-Achse

5.2.3 Typen und Sets

`TColorMode=(Normal, Red, Green, Blue, FColor);`

Möglichkeiten der Farbeinstellungen für Klasse TTImage.

`Normal`	Keine Farbanpassung
`Red`	24 Bit – Darstellung mit Blau- und Grünwert auf Null
`Green`	24 Bit – Darstellung mit Blau- und Rotwert auf Null
`Blue`	24 Bit – Darstellung mit Rot- und Grünwert auf Null
`FColor`	24 Bit – Falschfarbendarstellung

`KamSet=(Video, aSync, artificial);`

Einstellungsmöglichkeiten für Kameraaufnahmen

`Video`	Videomode zur Aufnahme (10 bis 10 000 ms)
`aSync`	ASyncmode zur Aufnahme (10 bis 10 000 µs)
`artificial`	Keine reale Aufnahme (z.B. durch Simulation erstellt)

5.2.4 Erstellte Klassen und Records

Header

Damit die Serienaufnahmen (*.TomoSerien) später allein stehend ausgewertet werden können, werden die Dateien mit einer Art nachgestelltem Header versehen. Dieser wird an die Datei später angehängt und enthält alle notwendigen Informationen zur genauen Rekonstruktion der Einstellungen. Der folgende Kodeabschnitt zeigt die Deklaration des Records:

```
Header = record                          //Anhang für Serienaufnahmen
                ID: Double;
                MessObject: String[255];
                Aufloesung: Byte;
                KamModus: KamSet;
                BelichtZeit: Word;
                ZeileDerDrehachse: Double;
                SumZeile: Byte;
                AusleseBereichOben: Byte;
                LinkeWL, RechteWL: Double;
                Bemerkung: String[255];
                MotorPos: Integer;
         end;
```

Die gespeicherten Informationen gleichen den in der Datenbank gespeicherten. Es wurde lediglich zur Vereinfachung die Länge der Stringfelder `MessObject` und `Bemerkung` auf 255 beschränkt. Die Größe eines abgespeicherten Records beträgt ca. 500 – 600 KB.

TMessDaten

```
TMessDaten=class
                        private
                                Iden: Double;
                                Mess: WideString;
                                Aufl: Integer;
                                KaMo: KamSet;
                                KaBe: Integer;
                                ZDDr: Double;
                                SumZ: Integer;
                                Libr: Integer;
                                ABeO: Integer;
                                LiWL: Double;
                                ReWL: Double;
                                Beme: WideString;
                                MoPo: TIntegerDynArray;
                                SpDa: TIntegerDynArray;
                                TodZ: Integer;
                                SaAl: Boolean;
                                procedure SetIdentifier(Value: Double);
                                procedure SetMessObjekt(Value: WideString);
                                procedure SetAufloesung(Value: Integer);
                                procedure SetKameraModus(Value: Kamset);
                                procedure SetKameraBelichtungszeit(Value: Integer);
                                procedure SetZeileDerDrehachse(Value: Double);
                                procedure SetLinienBreite(Value: Integer);
                                procedure SetSummierteZeilen(Value: Integer);
                                procedure SetAusleseBereichOben(Value: Integer);
                                procedure SetLinkeWL(Value: Double);
                                procedure SetRechteWL(Value: Double);
                                procedure SetBemerkung(Value: WideString);
                                procedure SetMotorPos(Index: Integer; Const Value: Integer);
                                procedure SetSpektralDaten(Index: Integer; Const Value:
                                    Integer);
                                function GetMotorPos(Index: Integer): Integer;
                                function GetSpektralDaten(Index: Integer): Integer;
                                function GetSpektralDatenCount: Integer;
                                function GetMotorPosCount: Integer;

                        public
                                property Identifier: Double read Iden write SetIdentifier;
                                property MessObjekt: WideString read Mess write SetMessObjekt;
                                property Aufloesung: Integer read Aufl write SetAufloesung;
                                property KameraModus: KamSet read KaMo write SetKameraModus;
                                property KameraBelichtungszeit: Integer read KaBe write
                                    SetKameraBelichtungszeit;
                                property ZeileDerDrehachse: Double read ZDDr write
                                    SetZeileDerDrehachse;
                                property SummierteZeilen: Integer read SumZ write
                                    SetSUmmierteZeilen;
                                property Linienbreite: Integer read Libr write
                                    SetLinienbreite;
                                property AusleseBereichOben: Integer read ABeO write
                                    SetAusleseBereichOben;
                                property LinkeWL: Double read LiWL write SetLinkeWL;
                                property RechteWL: Double read ReWL write SetRechteWL;
                                property Bemerkung: WideString read Beme write SetBemerkung;
                                property MotorPos[Index: Integer]: Integer read
                                    GetMotorPos write SetMotorPos;
                                property SpektralDaten[Index: Integer]: Integer
                                    read GetSpektralDaten write SetSpektralDaten;
                                property SpektralDatenCount: Integer read
                                    GetSpektralDatenCount;
                                property MotorPosCount: Integer read GetMotorPosCount;
                                property TodZeit: Integer read TodZ write TodZ;
                                property SaveAll: Boolean read SaAl write SaAl;
                                procedure MotorPosAdd(Value: Integer);
                                procedure MotorPosClear;
                                procedure SpektralDatenAdd(Value: Integer);
                                procedure SpektralDatenClear;
                                Constructor Create;
                                procedure UpDate();
                                procedure UpDateAusleseBereiche(Data: TStringList);
                                procedure SpektralDatenEliminateMulti;
                    end;
```

Diese Klasse speichert während des Programmablaufs die Daten des Programms. Sie bildet den zentralen Zugriffspunkt für alle anderen Programmteile, die aktuelle Einstellungsinformation benötigen. Ein einfacher Record reicht hier nicht aus. Es muss sichergestellt werden, dass keine ungültigen Daten aufgenommen werden können. Deshalb wurde hier auf die Möglichkeit von „`Property's`“ zurückgegriffen. Für den Benutzer der Klasse erscheinen die Felder wie Variablen. Die eigentlichen Werte werden allerdings in `private` Variablen gespeichert die von außen nicht zugänglich sind. Mit Hilfe von `read` und `write` Anweisungen werden die Daten übernommen. So werden die Werte klasseneigenen Proceduren und Funktionen übergeben. Diese überprüfen die eingehenden Werte bevor sie endgültig gespeichert werden. Gegebenenfalls werden zusätzlich Aktionen ausgelöst wie eine Aktualisierung der Oberfläche. Dies geschieht in der `procedure UpDate()`. Einige weniger interessante Zusatzfunktionen sollen lediglich das Arbeiten mit der Datenstruktur vereinfachen, fügen allerdings keine neue Funktionalität hinzu.
Während der Laufzeit existiert genau eine Instanz der Klasse.

TTImage

Wie der Name schon ausdrückt, handelt es sich um eine TImage – Komponente aus Delphi. Damit das Programm allerdings mit anderen Delphi – Installationen kompatibel bleibt, wurde darauf verzichtet, sie in die Komponentenleiste mit aufzunehmen. Deshalb muss sie auch manuell erstellt werden und kann nicht mit Drag&Drop auf die Oberfläche gezogen werden. Da sie von TImage abgeleitet wurde, ist die ursprüngliche Funktionalität weitgehend erhalten geblieben. Sie wurde allerdings durch oft benutzte Bildverarbeitungsaufgaben ergänzt.

```
TTImage=
        Class(TImage)
                public
                        constructor Create(AOwner: TComponent); override;
                        destructor Destroy; override;
                        procedure LoadFromSingleStream(FileName: String); overload;
                        procedure LoadFromSingleStream(Stream: TMemoryStream);
                                                                overload;
                private
                        Alpha: Boolean;
                        AlphaValue: Integer;
                        Inv: Boolean;
                        ColorMod: TColorMode;
                        MinG, MaxG: Integer;
                        MM: Boolean;
                        BBitmap: TBitmap;
                        KF100: array [0..199] of array [0..255] of byte;
                                    // speedup matrix Alphablend
                        procedure SetAlpha(Value: Boolean);
                        procedure SetAlphaValue(Value: Integer);
                        procedure SetColorMode(Value: TColorMode);
                        procedure SetInverted(Value: Boolean);
                        procedure PrepareKF100;
                        procedure AlfaBlendBmp24(DST, SRC: TBitmap;
                            X1, Y1, X2, Y2, W, H, K: Integer);
                        function F1_R(nr:Integer):Integer;
                        function F1_G(nr:Integer):Integer;
                        function F1_B(nr:Integer):Integer;
                        procedure SetMin(Value: Integer);
                        procedure SetMax(Value: Integer);
                        procedure SetMinMax(Value: Boolean);
                        procedure DoMinMax();
                published
                        property MinMax: Boolean read MM write SetMinMax;
                        property MinGrauWert: Integer read MinG write SetMin;
                        property MaxGrauWert: Integer read MaxG write SetMax;
                        property AlphaBlend: Boolean read Alpha write SetAlpha;
                        property AlphaBlendValue: Integer read AlphaValue write
                             SetAlphaValue;
                             //[0..99]~[undurchsichtig .. totaltransparent]
```

```
                              property Inverted: Boolean read Inv write SetInverted;
                              property ColorMode: TColorMode read ColorMod write
                                    SetColorMode;
                              procedure SetColor();
                              procedure AlfaBlend(SRC: TBitmap;
                                    X1, Y1, X2, Y2, W, H: Integer);
                              procedure BackUpOrgPicture();
                              procedure Invert(X,Y,W,H:Integer;
                                  Const OverrideOrg: Boolean = True);
                              Procedure KoordinatenSystem(Const Left: Double = 0;
                                  Const Right: Double = 255; Const dx: Double = 10;
                                  Const Abstand: Integer = 20;
                                  Const Color: TColor = clBlack);
                              procedure LoadFrom4096(Where: Pointer);
                              procedure Histogramm(Where: Pointer; Laenge: Integer; X, Y, W,
                                  H: Integer; MinWert, MaxWert: Integer; MinGrauWert,
                                  MaxGrauWert: Integer; Const Invert: Boolean=False);
                              procedure RestoreOrgPicture();
        end;
```

Die beiden Proceduren `LoadFrom4096` und `LoadFromSingleStream` erlauben es Bilder ohne Programmieraufwand von Festplatte bzw. aus dem Hauptspeicher zu laden. `LoadFrom4096` lädt ein Bild der Größe 640 x 480 12 Bit aus dem Arbeitsspeicher, skaliert dieses auf 256 Graustufen und stellt es dar. `LoadFrom-SingleStream` lädt ein gespeichertes aufgenommenes bzw. ausgewertetes Bild, skaliert es und stellt es mit 256 Graustufen dar. Die zusätzliche Procedure `Invert` erlaubt es, das gesamte Bild zu invertieren. `Histogramm` erlaubt es, eines der oft gebrauchten Histogramme aus einem Datensatz zu erstellen. `SetColorMode` erlaubt es den Bildinhalt in Falschfarben darzustellen. `AlphaBlend` bietet eine Möglichkeit mehrere Bilder zu überblenden. Die Procedure besteht hauptsächlich aus Assemblercode und sollte deshalb nur von erfahrenen Programmierern verändert werden, da bei Fehlern Auswirkungen auf das Betriebssystem zu erwarten sind.

TMyCustomPanel

Hilfsklasse für Formular “Bildbereiche auswählen”. Diese Klasse erleichtert die Auswahl der Spektrallinien. Eine genauere Erläuterung erfolgt in den folgenden Kapiteln.

```
        TMyCustomPanel =  Class
                              Panel1:TPanel;
                              Panel2:TPanel;
                              Panel3:TPanel;
                              NP:TPoint;
                              OS: Integer;
                              Orientation: GOrientation;
                              Constructor Create(Owner: TWinControl; Nullpkt: TPoint;
                                  Offset: Integer; Const POrientation: GOrientation =
                                  aVertical; Const Visible: Boolean = True);
                              Destructor Destroy();override;
                              function getDaten:TStrings;
                              procedure BreiteSetzen(Offset: Integer);
                              procedure BringToFront();
                              procedure verschieben(X: Integer);
                              function Sort(Item1, Item2: Pointer): Integer;
                              procedure MoveTo(X: Integer);
                              procedure Hide();
                              procedure Show();
                        end;
```

5.3 Die Datenbank „Tomografische Emissionspektroskopie“

Die aufgenommenen und ausgewerteten Daten werden durch das Programm in einer XML – Datei gespeichert. Um die Datenbankgröße zu minimieren, wurden jedoch die durch die Kamera aufgenommenen sowie durch die Tomografie ausgewerteten Daten selbst, in einzelne Dateien ausgelagert. Auf diese Weise steht mehr Arbeitsspeicher für die speicherintensive Auswertung zur Verfügung. Die XML (Extensible Markup Language) Sprache ähnelt sehr stark der weit verbreiteten HTML Sprache, die für Internetseiten eingesetzt wird. Sie kann genauso wie diese mit einem einfachen Editor betrachtet und verändert bzw. mit einem Webbrowser angezeigt werden.

```
<Datenbank>
 <DatenSatz Identifier="2452551,9440667" MessObjekt="Simulation"
  Aufloesung="180" KameraModus="artificial" KameraBelichtungszeit="0"
  ZeileDerDrehachse="0" SummierteZeilen="0" Linienbreite="0" Auslese-
  BereichOben="0" LinkeWL="0" RechteWL="-1" Bemerkung="Simulation
  vom 04.10.2002 10:39:27">
  <Einzelmessung MotorPos="0" SpektralDaten="0" Iterationsschleifen="10" Auf-
   nahmeDaten="MessDaten\2452551.94406674.0.0" AusgewerteteDa-
   ten="MessDaten\2452551.94406674.0.0.DONE" />
   ...
 </DatenSatz>
   ...
</Datenbank>
```

Abbildung 5-4: Gliederung XML – Datei „Datenbank.xml“ angezeigt durch Microsoft Internet Explorer Version 6 (Bsp. Simulierte Messung)

Die Struktur besteht aus nur 3 Tags: <Datenbank>, <DatenSatz> sowie <Einzelmessung>. Der Tag <Datenbank> bildet hier die Dokumentenwurzel (document root). Es kennzeichnet den Beginn und das Ende des gesamten Dokuments.
Der Tag <DatenSatz> schließt genau eine vollständige Aufnahme also eine gesamte Messung mit allen Spektrallinien und Motorpositionen ein. Er enthält somit stets mindestens einen Tag <Einzelmessung>, der eine Messung repräsentiert (eine Spektrallinie an einer bestimmten Motorposition).
Der Tag <DatenSatz> enthält folgende Attribute:

- Identifier: Aufnahmedatum im julianischen Format
- MessObjekt: Name des Messobjektes
- Aufloesung: Auflösung der Messung
- KameraModus: Kameraaufnahmemodus (aSync, Video, aritficial)
- KameraBelichtungszeit: Eingestellte Belichtungszeit
- ZeileDerDrehachse: Festgestellte Zeile der Drehachse
- SummierteZeilen: Anzahl der summierten Zeilen
- Linienbreite: Breite der aufgenommenen Spektrallinien
- AusleseBereichOben: obere Pixelposition des horizontalen Bereiches
- LinkeWL: Linke detektierte Wellenlänge
- RechteWL: Rechte detektierte Wellenlänge
- Bemerkung: optionale Bemerkung zur Messung

Im leeren Tag <Einzelmessung> müssen jetzt nur noch wenige veränderliche Werte für jeden Bereich der Messung gespeichert werden:

- MotorPos: horizontale Motorposition während der Messung
- Spektraldaten: linke Grenze der aufgenommenen Wellenlänge
- Iterationsschleifen: Anzahl der Iterationsschleifen bei der Auswertung
- AufnahmeDaten: rel. Pfad der Messdatei bezogen auf Prog.
- AusgewerteteDaten: rel. Pfad der ausgewerteten Datei bez. auf Prog.

Die Pfadangaben im Tag <Einzelmessung> setzen sich hierbei wie folgt zusammen:

- Relativer Speicherpfad bezogen auf den Pfad des Programms selbst
  in Abbildung 5-4 – 1 z.B. MessDaten\
- Aufnahmezeitpunkt im julianischen Datumsformat
  in Abbildung 5-4 – 1 z.B. 2452551.94406674 ≈ 04.10.2002 10:39:27
- Motorposition der Aufnahme
  in Abbildung 5-4 – 1 z.B. 0
- Linke Grenze der aufgenommenen Wellenlänge
  in Abbildung 5-4 – 1 z.B. 0

Zur Kennzeichnung einer ausgewerteten Datei wird noch das Kürzel „DONE“ angehängt.
Eine eindeutige Zuordnung der gespeicherten Datensätze wird somit durch die Angabe des julianischen Datums[3] und die Unterscheidung nach lokalem und spektralem Aufnahmeort sichergestellt.
Bei jeder der gespeicherten, aufgenommenen bzw. ausgewerteten Dateien handelt es sich um durch Delphi gespeicherte vier Byte - Streams[4].

Delphi stellt mit der Klasse TXMLDocument eine nichtvisuelle Komponente zur Verfügung, die in der Lage ist das vorhandene XML – Files zu öffnen, speichern sowie zu analysieren. Die enthaltenen Daten werden in einer arrayähnlichen Datenstruktur zur Verfügung gestellt und können so auf einfachste Weise bearbeitet werden. Die Schnittstelle zur XML – Datei muss hierfür nur einmal mit Hilfe des „XML Data Binding Wizard“ hergestellt werden.
Dieser erstellt nach kurzer Konfigurationsphase eine Delphiunit, die vollen Zugriff auf die XML – Datei ermöglicht. Die Unit definiert gemäß den Tags der XML – Datei drei Typen:

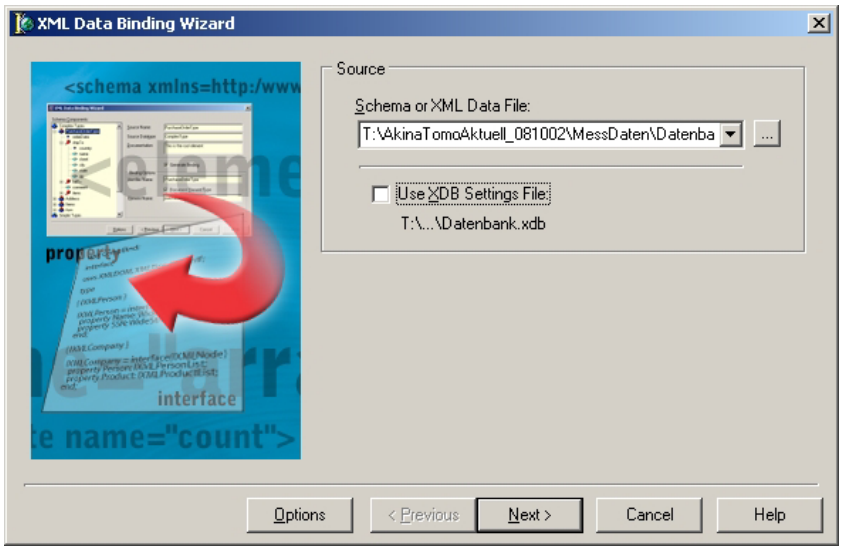


Abbildung 5-5: XML Data Binding Wizard

- `IXMLDatenbankType`
- `IXMLDatenSatzType`
- `IXMLEinzelmessungType`

Diese dienen als Grundlage für die drei angelegten Klassen, die sich auf die drei Typen abstützen. Die Klassen regeln alle Zugriffsrechte sowie die Typen der gespeicherten Information. Die Daten werden bei Zuweisung nicht direkt, sondern mit

[3] Julianisches Datum: Anzahl der Tage (einschließlich Tagesbruchteile) seit dem 1. Januar 4713 vor Christus, Greenwich, Mittag (eindeutiges auf Zählung der Tage basierendes Datierungssystem)

[4] Stream: bitorientierter dynamischer Speicherbuffer mit erweiterter E/A – Funktionalität ohne wahlfreien Zugriff

Hilfe von Properties erst nach einer Überprüfung (nähere Informationen siehe Delphi – Hilfe).
Um die Datei zu laden bzw. neu zu erstellen werden neben den drei Haupttypen noch drei Funktionen angelegt:

- `function GetDatenbank(Doc: IXMLDocument): IXMLDatenbankType;`
- `function LoadDatenbank(const FileName: WideString): IXMLDatenbankType;`
- `function NewDatenbank: IXMLDatenbankType;`

Um einen neuen Datensatz korrekt zu übernehmen werden zusätzlich drei Funktionen benötigt:

- `function NeuerEintragDatenBank(Identifier: TDateTime; MessObjekt: String; Aufloesung: Integer; KameraModus: KamSet; KameraBelichtungszeit: Integer; ZeileDerDrehachse: Double; SummierteZeilen, Linienbreite:Integer; LinkeWL, RechteWL: Single; AusleseBereichOben: Integer; Bemerkung: String): IXMLDatenSatzType; overload;`
- `function NeuerEintragDatenBank(Identifier: TDateTime; MessObjekt: String; Aufloesung: Integer; KameraModus: String; KameraBelichtungszeit: Integer; ZeileDerDrehachse: Double; SummierteZeilen, Linienbreite:Integer; LinkeWL, RechteWL: Single; AusleseBereichOben: Integer; Bemerkung: String): IXMLDatenSatzType; overload;`
- `function NeuerUnterEintragDatenBank(ParentNode: IXMLDatenSatzType; MotorPos, SpektralDaten: Integer; AufnahmeDaten: TMemoryStream; DatName: String):IXMLDatenSatzType;`

Die globalen Funktionen speichern durch einen einzigen Aufruf einen kompletten Datensatz und erleichtern so die Speicherung, um Zuweisungsfehler und somit die Zerstörung der Datenbank zu verhindern.
Eine Komponente vom Typ `TXMLDocument` übernimmt die Speicherung und Verwaltung. Wichtig dabei ist, dass eine Aktualisierung der Datei nur dann stattfindet, wenn mit Hilfe der Komponente eine Speicherung durchgeführt wurde. Sonst werden die Änderungen bei Programmende verworfen.

Neben der Datenbank existieren noch wenige andere Formate, die von der Anwendung benutzt bzw. gespeichert werden:

- *.TomoBilder (Speicherdump eines 12Bit Bildes – Größe: 600 KB)
- *.TomoSerien (Speicherdump aller 12Bit Bilder einer 180° Drehung – Größe: ~18 – ~108 MB
- Spektraldaten.ini (wichtige Spektrallinien verschiedener Betriebsgase)
- \SpektraData*.XML (tabellierte Spektrallinien verschiedener Betriebsgase)
- Einstellungen.XML (gespeicherte Messeinstellungen)
- *.bmp (verschiedene Screenshots)
- *.PositionsListe (ini – Datei für Schrittmotorsteuerung)
- *.CheckList (Checklistdateien)
- *.ITM (Datendateien für Simulation)

5.4 Justage und Datenaufnahme

5.4.1 Das Formular „Schrittmotorsteuerung“

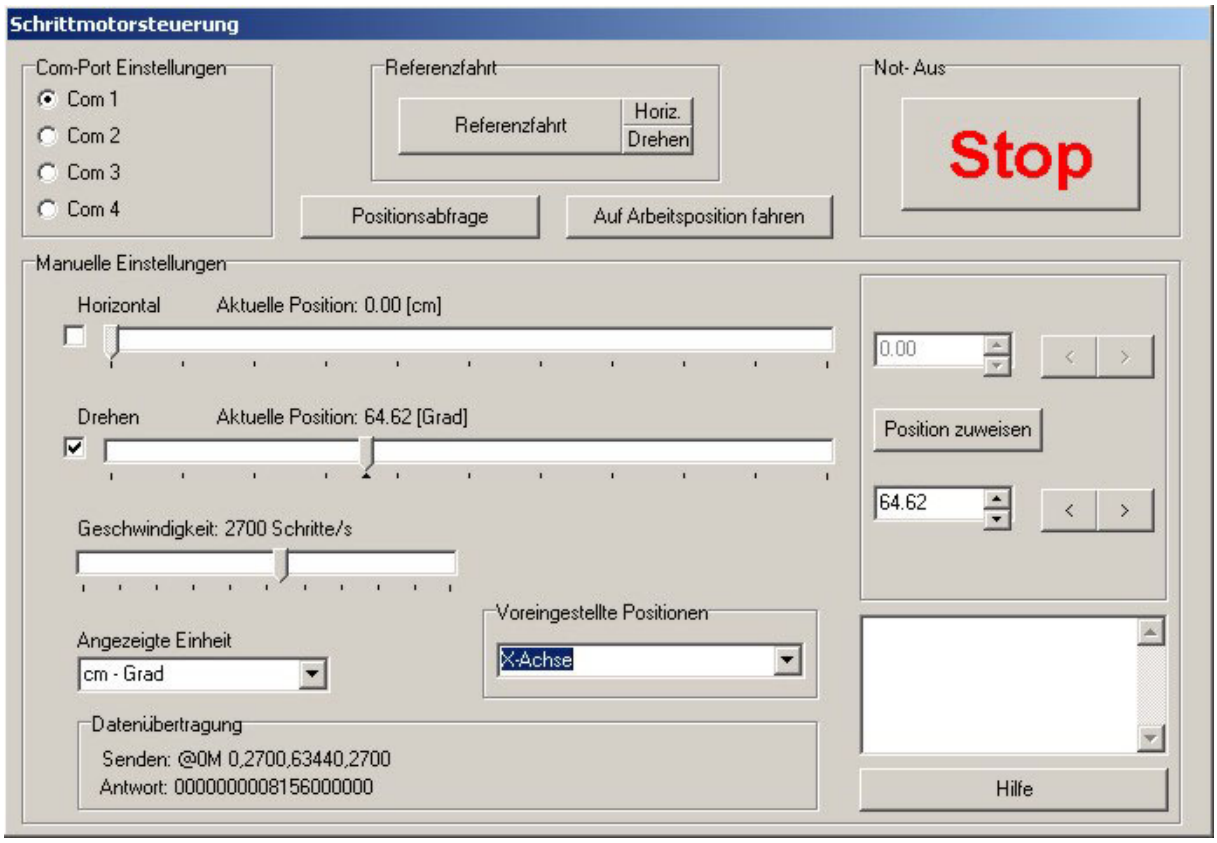


Abbildung 5-6: Das Formular "Schrittmotorsteuerung"

Beschreibung:
Dieses Formular bildet die Schnittstelle zu den beiden für Fahr- bzw. Drehbewegung eingesetzten Schrittmotoren. Es bietet einige Möglichkeiten, sie, an die jeweiligen Ansprüche angepasst, anzusteuern.

Der Ablauf der bei einem Neustart des Programms vorzunehmenden Einstellungen:

1. Auswahl des Com- Ports, an dem die Steuerung angeschlossen ist.
2. Durchführen einer **Referenzfahrt** oder einer **Positionsabfrage.**

Worauf muss geachtet werden:
Bei jeder Bewegung muss darauf geachtet werden, dass die Apparatur nirgends anstößt und blockiert. Die Schrittmotoren haben keinen Überlastschutz und können so bleibenden Schaden davontragen. Des weiteren ist zur exakten Drehbewegung zwischen Motor und Trommel ein Getriebe mit großer Untersetzung (100:1) angeschlossen. Bei Anstoßen der Trommel reicht das dadurch verstärkte Drehmoment des Motors aus, die Achse abzudrehen!!
Im normalen Betrieb stellt die Software bestmöglich sicher, dass die Grenzpositionen nicht erreicht werden können, aber gerade bei der Referenzfahrt kommt es darauf an, dass alle Endschalter einwandfrei funktionieren; deshalb hier immer mit besonderer Vorsicht vorgehen und regelmäßig den sicheren Kontakt aller Kabelverbindungen überprüfen!

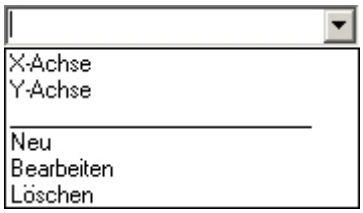
Die verschiedenen Eingabemöglichkeiten:

- Track-Bars:
  Zielposition einstellen, dann Position Zuweisen drücken
- Edit- Felder:
  Zielposition in Schritten bzw. cm/grad angeben, Position Zuweisen drücken
- Pfeiltasten:
  So lange drücken, bis gewünschte Position erreicht.
  Achtung: Bei Geschwindigkeit > 3500 Schritte/s ist es möglich, dass bei Loslassen der Motor weiterfährt (Ursache des Problems ungeklärt, hängt mit Datenübertragung zwischen Rechner und Interface zusammen), dann mehrmals STOP betätigen!
- Memo-Box:
  Über schriftliche Eingaben kann zum Beispiel angegeben werden, wie weit die Messanordnung in eine bestimmte Richtung vom Ausgangspunkt aus bewegt werden soll.
  z.B.: Y -10.gr Y : Drehen
  -10 : um –10
  .gr : Grad ausgehend von der jetzigen Position
  siehe hierzu die Hilfe
- Voreingestellte Positionen:
  Wichtige, oft gebrauchte Positionen können hier abgespeichert oder aufgerufen werden.

Voreingestellte Positionen benutzen:

Gerade zu Einstellungszwecken (zum Beispiel zur Einrichtung der Optik) muss die Messanordnung oft zu definierten Positionen gefahren werden. Um sich die jeweilige Anzahl an Schritten nicht merken zu müssen, wurde die Möglichkeit geschaffen, diese zu speichern.

Die hierfür verwendete ComboBox stellt gleichermaßen Menüs zum Auswählen bereits gespeicherter Positionen, als auch zum Erstellen neuer Positionen zur Verfügung.



Wichtig für die Speicherung der Daten ist die INI-Datei *PosList.Positionsliste* im Hauptverzeichnis der Akina-Software.
Fehlt diese, so wird sie neu erstellt und enthält zwei Voreinstellungen:

X-Achse: Drehposition zur optimalen Einrichtung des Messobjektes auf der X-Achse (horizontal)
Y-Achse: Drehposition zur optimalen Einrichtung des Messobjektes auf der Y-Achse (vertikal)

Diese Voreinstellungen basieren darauf, dass die durch die Spiegel eingestellte Drehachse exakt auf der tatsächlichen Drehachse in der Trommelmitte liegt. Sollte sich daran etwas ändern, so muss man die Konstanten `EinstX` und `EinstY` in `Globals` ändern.
Ein Klick auf Neu, Bearbeiten oder Löschen öffnet eine während der Laufzeit erstellte Form. Realisiert ist dies in der Prozedur
`BearbeitenFenster(var Value: TPosition)` (öffnet den Dialog zur Bearbeitung von Positionsdaten) und der

Funktion `Auswahlfenster(Eintraege: TStringList): Integer` (öffnet den Dialog zur Auswahl eines gespeicherten Eintrages und gibt nach Betätigen des OK-Buttons dessen Index zurück). Beide sind in der Klasse `TPosVorgab` in der `Motorsteuerung` zu finden.

Zum Aufbau der Befehle für das Interface:
Um die einzelnen Funktionen des Interface ansprechen zu können, wird ein String über den Com-Port verschickt:

Initialisierung:	'@03'	hier Initialisierung auf 2 Achsen
Positionsabfrage:	'@0P'	
Nothalt:	CHR(255)	
XReferenzfahrt:	'@0R1'	Fahrbewegung
YReferenzfahrt:	'@0R2'	Drehbewegung
Referenzfahrt allgemein:		
	'@0d <GeschwindigkeitX>,<GeschwindigeitY>'	

Befehl zum Anfahren einer bestimmten Schrittposition:
'@0M <ZielpositionX>, <GeschwX>, < ZielpositionY>, < GeschwY>'

Die Befehle werden von der **`procedure`** `Befehl_Senden(S: String);` an den Com-Port-Treiber weitergeleitet.

Prinzipbild des Anschlusses des Interface an den Computer:

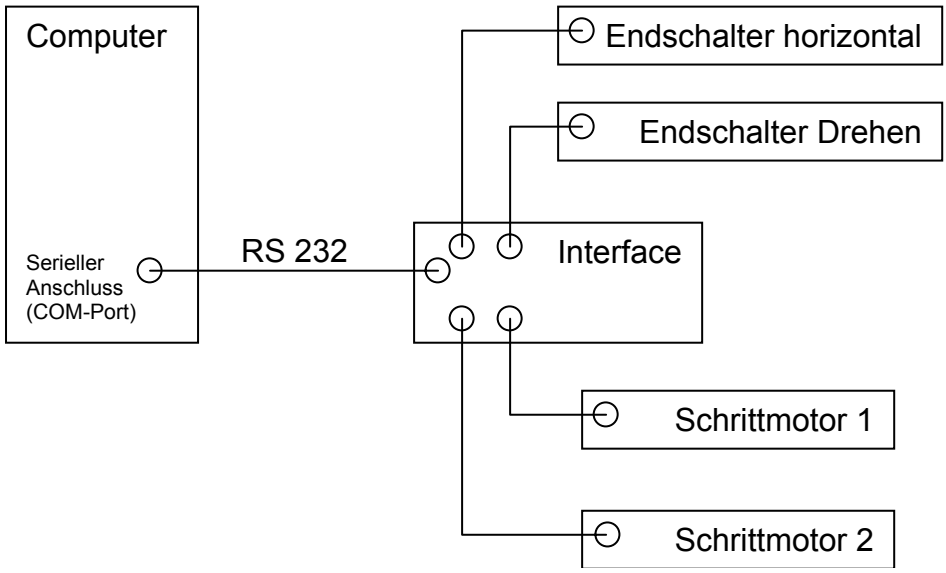


Abbildung 5-7: Anschluss des Interface an den Computer

5.4.2 Das Formular „Kamera-Livebild“

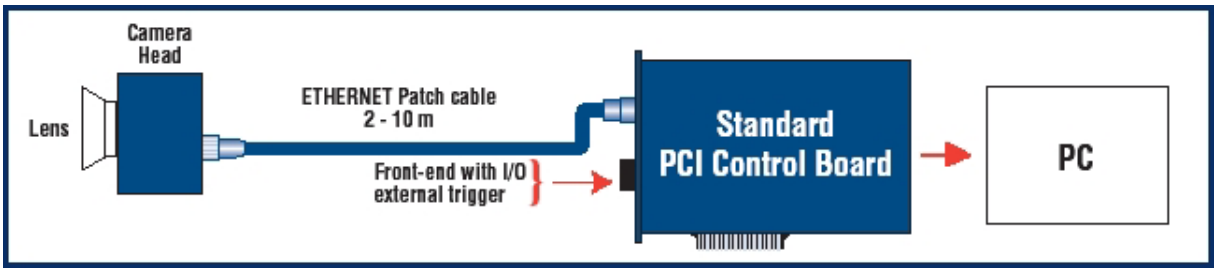


Abbildung 5-8: Kameraprinzip

Zur Darstellung des Kamerabildes ist externe Hardware nötig. Dies ist in Abbildung 5-8 dargestellt.

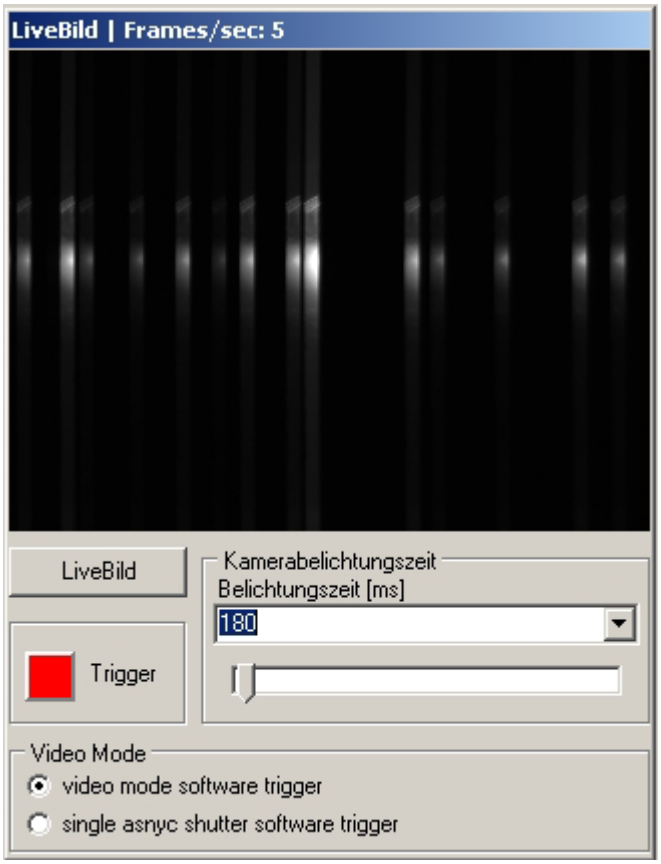


Abbildung 5-9: Das Formular Livebild

Um stets ein aktuelles Kamerabild zur Verfügung zu stellen, wurde dieses Formular erstellt. Hier wird auf einer Größe von 320 x 240 Pixel das Kamerabild auf die Hälfte verkleinert dargestellt. In der Titelleiste ist die Anzahl der dargestellten Bilder pro Sekunde ablesbar. Sobald ein neues Bild aufgenommen ist, wechselt zusätzlich das Feld „Trigger“ seine Farbe, um dies zu verdeutlichen.
Die in diesem Formular angezeigten Bilder werden nicht im Hauptspeicher des Computers abgelegt, sondern dienen nur dazu, um ein Monitorbild darzustellen.
Das wichtigste an diesem Formular ist, dass sich hier die Belichtungszeit sowie der Kameramodus einstellen lassen.

Man kann hier zwei verschiedene Aufnahmeverfahren unterscheiden:

- „Video mode software trigger“
- „single aSync shutter software trigger”

Der für uns relevante Unterschied ist hier nur die Belichtungszeit. Im „Videomodus“ ist sie einstellbar von 10 bis 10 000 ms, Im „aSync Shutter“ Modus von 10 bis 10 000 µs. Deshalb findet der letztere vor allem bei starken Lichtquellen, der Videomodus nur bei schwach strahlenden Messobjekten, seine Anwendung. Die Einstellung erfolgt entweder über den im Feld Kamerabelichtungszeit befindlichen `Trackbar` mit der Maus bzw. über die sich dort befindliche `ComboBox` mit Tastatur. Zusätzlich können aus der `ComboBox` noch markante voreingestellte Werte ausgewählt werden.

Die hier durchgeführten Einstellungen der Belichtungszeit und des Kameramodus sind für das gesamte Programm gültig. Auf Grund dessen, müssen sie vor jeder Messung kontrolliert bzw. angepasst werden.

Zur Realisierung des Livebildes wird beim Start der Hauptanwendung ein eigener Thread erstellt. Dieser läuft mit niedriger Priorität. Auf diese Weise wird eine beeinträchtigung des Hauptprogramms verhindert, denn das Livebild dient nur zur oberflächlichen Kontrolle des Kamerabildes.

```
Constructor TLiveBildThread.Create(Suspended: Boolean);
begin
        inherited Create(Suspended);
        Priority:=tpLowest;
end;
```

Anschließend wird die Kamera wie im Anhang 7.3 beschrieben initialisiert und Speicher reserviert.
Ist das Fenster sichtbar, nimmt der `TLiveBildThread` ein Bild auf und stellt dieses verkleinert auf dem Bildschirm dar. Dazu werden die aufgenommenen 12 Bit – Werte nicht dauerhaft gespeichert, sondern direkt auf 256 Graustufen skaliert und in einer Image - Komponente dargestellt. Der Aufruf der hierfür notwendigen Funktionen muss als `Synchronized` erfolgen, da die Anwendung selbst in einem anderen Thread läuft und Zugriffskonflikte entstehen könnten, die einen Absturz zur Folge hätten.
Benötigt nun ein anderes Formular ein aktuelles Bild, wird der Thread dazu aufgefordert, ein Bild in die globale Variable BildDaten12bit zu speichern. Dabei handelt es sich um einen Zeiger auf ein zweidimensionales Array der Größe 640 x 480 vom Typ Word:

```
BildDaten12bit: PBild;
PBild = ^TBild;
TBild = array[0..AusWerteBildHeight-1,0..AusWerteBildWidth-1] of Word;
```

Das Bild soll allerdings nur von dem Formular übernommen werden, von dem es auch angefordert wurde. Deshalb wurden zwei Variablen definiert:

- `GetscreenShot`
- `DrawScreenShot`

```
TMaleFunction = procedure of Object;
GetScreenShot: Boolean;
DrawScreenShot: TMaleFunction;
```

Ist im Programm die Variable `GetScreenShot` auf `True` gesetzt, wird eine Aufnahme in das Array `BildDaten12bit` übertragen. Anschließend wird der Wert automatisch wieder zurückgesetzt. Soll das Formular ebenfalls aktualisiert werden, muss die Variable `DrawScreenShot` gesetzt werden. Es handelt sich dabei um einen Zeiger auf eine Methode des Programms. Das Formular kann somit hier einen Zeiger auf die Methode übergeben, die später die Weiterverarbeitung der Daten übernimmt. Sobald das Bild aufgenommen wurde und abgelegt ist, wird diese Funktion aufgerufen. Die entstehende Wartezeit muss somit nicht mit Busy – Waiting überbrückt werden. Dies hätte wieder ein Blockieren der Anwendung zur Folge.
Bleibt der Wert von `DrawScreenShot` auf `nil`, erfolgt kein anschließender Funktionsaufruf. Es wird lediglich das Bild in den Hauptspeicher übertragen.

Das Formular führt die eigentliche Überwachung der Kameratemperatur durch. Dazu wurde ein `Timer` verwendet. Alle fünf Sekunden wird durch Aufruf der Funktion `READTEMPERATURE` der eingebaute Temperatursensor der Kamera abgefragt. Betriebsbereit ist die Kamera von –55 °C bis +125 °C. Um eine Beschädigung der Kamera zu vermeiden, wird bei Erreichen von 100 °C eine Warnung ausgelöst. Ist diese

Temperaturgrenze überschritten, wird aus Sicherheitsgründen das Timerintervall auf 500 ms heruntergesetzt, bis die Temperatur wieder gesunken ist. Die Betriebstemperatur der Kamera pendelt sich auf etwa 50 °C ein. Dies geschieht schon wenige Minuten nach Einschalten des Computers. Wie erste Tests ergeben haben, erfolgt auch bei geschlossener Kameraschutzbox und laufenden Plasmabrennermessungen keine Erhöhung der Temperatur. Es sind deshalb auch keine Beeinträchtigungen der Messung auf Grund von Temperaturschwankungen der Kamera zu erwarten.

Ein während der Entwicklung nicht zu lösendes Problem stellt die Verzögerung beim Umstellen der Belichtungszeit dar. Es kann nicht gewährleistet werden, dass eine Neueinstellung der Belichtungszeit erfolgreich war. Um die gewünschte Belichtungszeit dennoch sicher einzustellen wird überprüft, ob die Einstellung korrekt übernommen wurde. Ist dies nicht der Fall, wird die Einstellung wiederholt, bis die gewünschte Belichtungszeit gesetzt werden konnte. Dies kann zu Verzögerungen im Sekundenbereich führen.

5.4.3 Das Formular „Aufnahme“

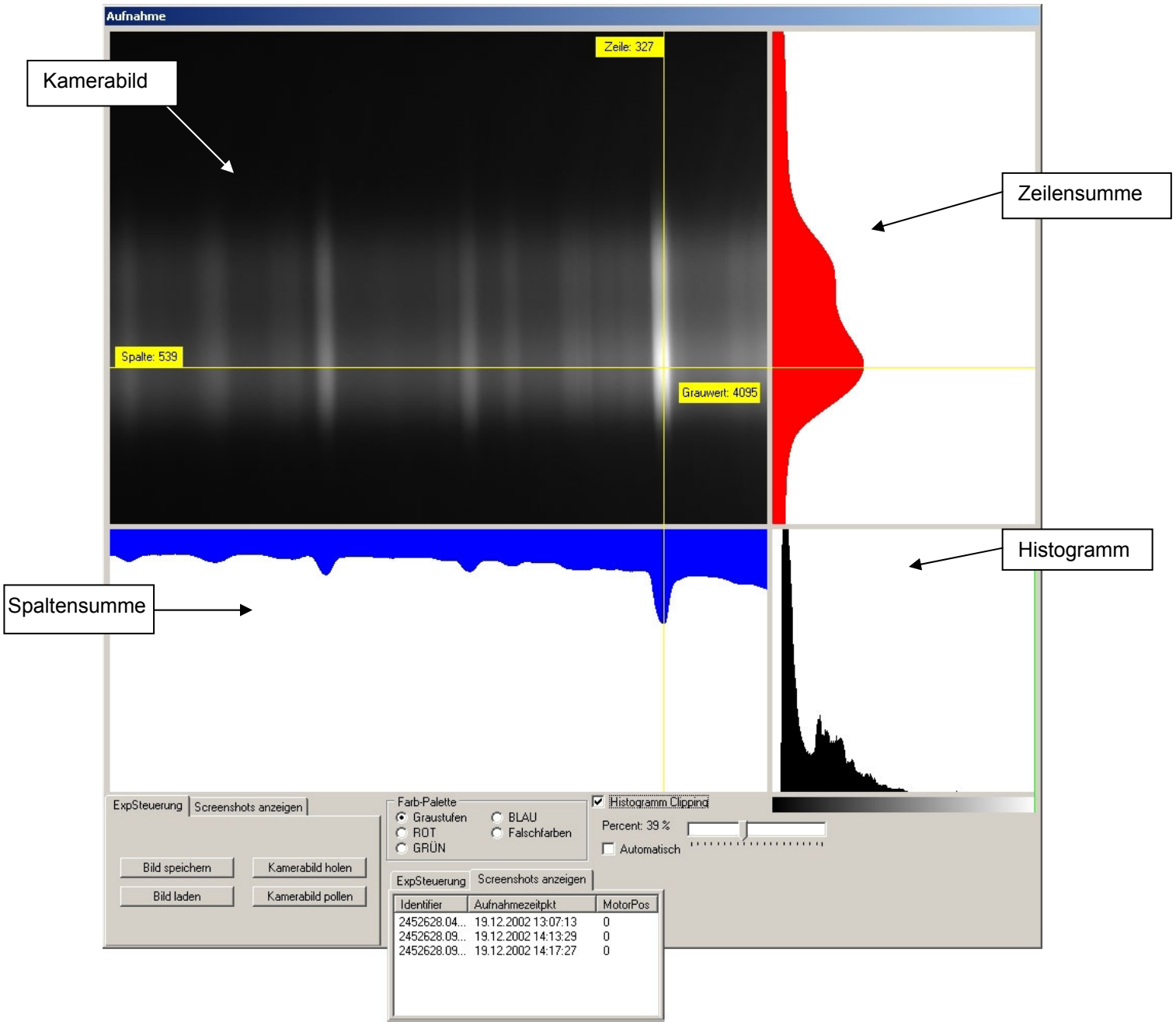


Abbildung 5-10: Das Formular Aufnahme

Abbildung 5-10 zeigt die Oberfläche des Formulars Aufnahme. Man benötigt dieses Formular vor allem für Kalibrieraufgaben. Es ist in der Lage ein aufgenommenes Bild (Livebild oder gespeichertes Bild) anzuzeigen. Zusätzlich werden die Spalten- bzw. Zeilensummen dargestellt. Hier lassen sich ganz leicht Dejustierungen der Kamera, des Spektrographen bzw. der Messanordnung feststellen. Man kann diese an der Form der Spaltensumme erkennen. Ergeben sich dort keine scharfen Rechtecke, muss eine Nachjustierung vorgenommen werden.
Durch Bewegen der Maus über die einzelnen Bilder, werden zusätzliche Informationen wie Grauwert bzw. Spalten-, Zeilensumme und aktuelle Pixelposition der Maus angezeigt.

Das Histogramm rechts unten ermöglicht außerdem eine Übersicht über die jeweiligen Grauanteile des dargestellten Bildes. Hierbei lässt sich die Anzeigeschwelle durch Benutzung der Funktion „Histogramm Clipping“ verstellen. Die Funktion „Automatisch“ stellt eine Normierung auf die zweitstärkste Linie dar.
Durch Änderung der Farb – Palette können sehr schwache Linien bzw. überbelichtete Linien erkannt werden.
Die Schaltflächen „Bild laden“ und „Bild speichern“ erlauben das Abspeichern und spätere Laden einzelner Kamerabilder. Das Dateiformat hierfür ist eine typisierte Datei in der sequenziell die 640 x 480 aufgenommenen „Word[5]“ – Werte gespeichert werden. Sie trägt die Endung „*.TomoBilder“.
Die Schaltfläche „Kamerabild holen“ nimmt genau ein Kamerabild auf. Die Schaltfläche „Kamerabild pollen“ nimmt jede Sekunden ein Bild auf und stellt dieses dar. Auf diese Weise kann leicht eine Kalibrierung stattfinden.
Die Umschaltung auf die Karteikarte „Screenshot anzeigen“ führt zur Anzeige der im Verzeichnis „\MessDaten“ liegenden Screenshots, die während der Messung automatisch aufgenommen werden. In der dargestellten Liste lassen sich dazu Aufnahmezeitpunkt und Aufnahmeort ablesen.

Die im Hauptformular eingestellten Werte für Bildinvertierung oder Beschneidung auf einen kleineren Farbbereich werden von diesem Formular berücksichtigt.

[5] Word: Bereich: 0..65535 (unsigned 16-bit)

5.4.4 Das Formular „Drehachsen-Bestimmung“

Vorbemerkung:
Elementar für die Qualität der tomografisch rekonstruierten Schnittbilder ist die Kenntnis der exakten Drehachsenposition (deren Zeilennummer) im aufgenommenen Kamerabild.
Diese Drehachsenposition kann experimentell sehr genau und ohne großen Aufwand ermittelt werden.
Im Software - Projekt ist dafür ein eigenes Formular vorgesehen, dessen Funktionalität nachfolgend beschreiben wird:

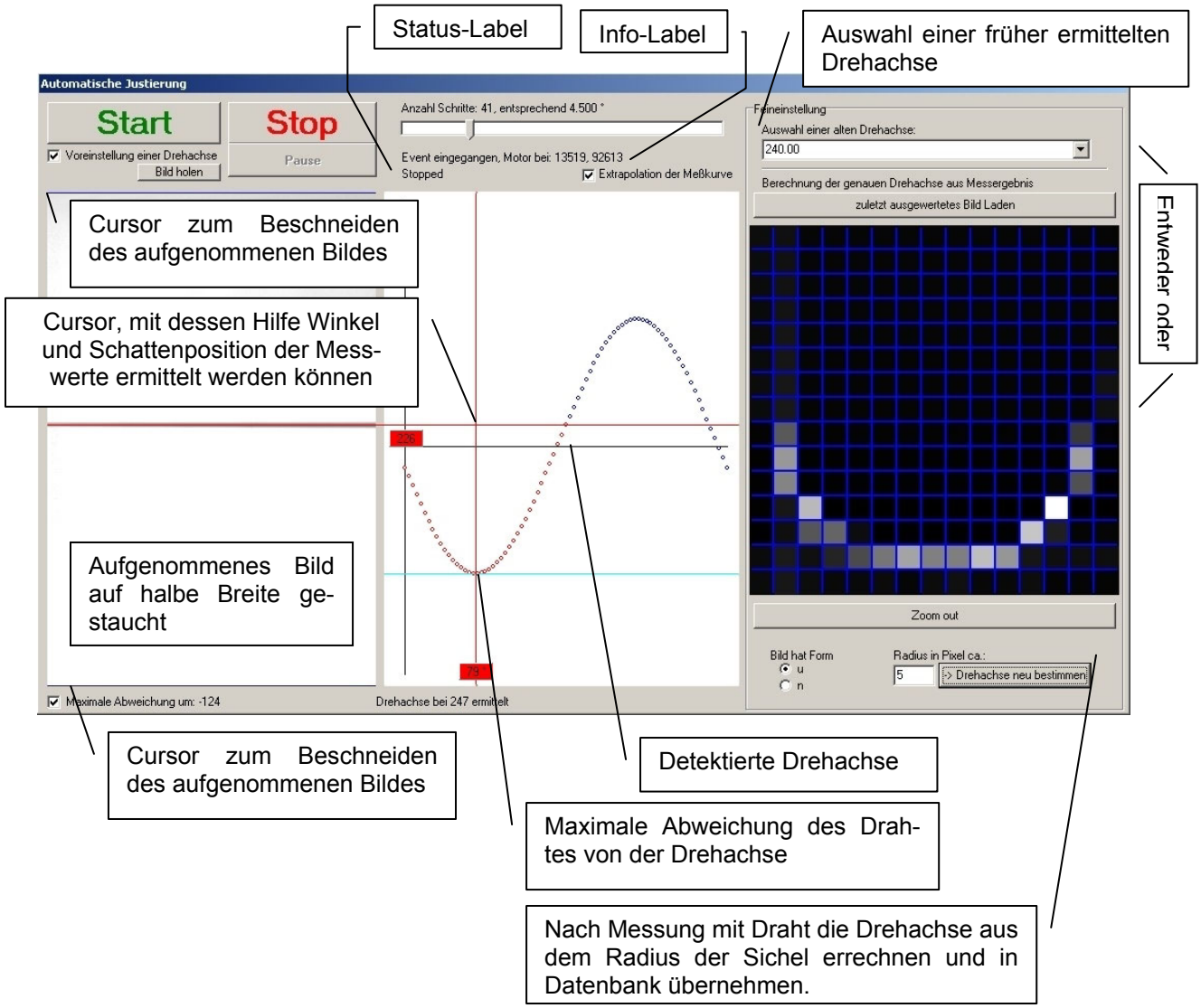


Abbildung 5-11: Das Formular "Drehachsen-Bestimmung"

Beschreibung:
Das Formular „Drehachsenbestimmung“ soll als Hilfsmittel zur Justierung der Drehachse und zur Ausrichtung des Messobjektes für die Messung dienen.
Die Durchführung erfolgt zunächst automatisch, es wird ein dünner Draht und eine möglichst ideale Weißlichtquelle benötigt.

Automatische Justierung:
Vorbereitend muss ein Draht im Aufnahmebereich der Kamera befestigt werden. Mit Hinterleuchtung sollte im Livebild (Einstellung 110ms, Video-Mode) ein deutlicher dunkler Schatten auf hellem Hintergrund zu sehen sein.
Um ein aussagekräftiges Messergebnis erzielen zu können muss nun der Draht so befestigt werden, dass sein Schatten bei einer Drehung der Trommel um 180° stets im Sichtbereich der Kamera bleibt.
Der letzte Schritt vor Beginn der Messung ist die Einstellung der Schrittweite an der dafür vorgesehenen TrackBar. Je geringer die Schrittweite gewählt wird, um so exakter aber auch langsamer ist die Messung. Ein guter und ausreichender Wert ist die Voreinstellung von 19 Schritten, entsprechend 10°.
Bei Klick auf START setzt sich jetzt die Trommel in Bewegung und die Messung läuft ab. Mit STOP kann jederzeit die Messung abgebrochen werden, was zur Folge hat, dass alle bereits ermittelten Messwerte verworfen werden. Soll dies nicht passieren, steht hierfür der Button PAUSE zur Verfügung.

Feineinstellung/ Voreinstellung einer alten Drehachse:
Die automatische Justierung liefert durch die Messart bedingt eine nicht hundertprozentige Angabe über die Position der Drehachse. Deshalb sollte man nach einer ersten groben Drehachseneinstellung eine Messung durchführen, bei der möglichst viele Zeilen zusammengefasst werden (siehe „Das Formular Bildbereich Auswählen“). Man erreicht damit, dass der Draht als punktförmige Lichtquelle in der Größe eines Pixels auftritt und der Tomografie-Algorithmus am genausten arbeitet. Zeigt sich hierbei statt des erwarteten Pixels ein sichelförmiges Gebilde (siehe Abschnitt „Überprüfung der Genauigkeit und Wichtigkeit der Daten der Drehachsenbestimmung“) im ausgewerteten Bild, so ist eine Nachbesserung der Drehachse von Hand notwendig.
Das Formular bietet dafür eine komfortable Hilfe:
Zunächst einmal CheckBox Voreinstellung einer Drehachse auswählen anklicken, dann auf zuletzt ausgewertetes Bild klicken und das sichelförmige Gebilde heranzoomen. Nun dessen Radius in Pixeln abschätzen (siehe hierzu folgende Tabelle) und in das EditFeld eintragen. Nach Bestätigung wird die neue Drehachse berechnet und für weitere Messungen übernommen.

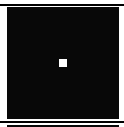
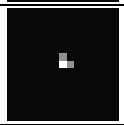
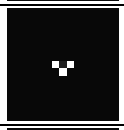
gezoomtes Bild	Radius der Sichel (ca. - Angabe)
	0
	0, kann erfahrungsgemäß nicht weiter verbessert werden
	0,7
	1,3
	5

Abbildung 5-12: Übersicht über verschiedene Sichelgrößen und den zugehörigen Einstellungen

Sollte dieser Effekt bei Messwerten auftreten, deren Messung nicht wiederholt werden kann, so gibt es die Möglichkeit, mit Hilfe des Formulars „MessDatenAufarbeitung“ nachzubessern.

Die Messergebnisse:
Nach Ablauf der Messung stehen zwei wichtige Messergebnisse zur Verfügung: Die Position der Drehachse und die maximale Abweichung des Drahtes von der Drehachse, jeweils in Pixel.
Idealerweise sollte die Drehachse exakt in der Mitte des Aufnahmebereichs der Kamera liegen, also bei Zeile 240 (Auflösung vertikal: 480). Die Drehachse kann mit Hilfe der beiden Spiegel eingestellt werden.

Die Abweichung des Drahtes ist in gewissen Grenzen tolerierbar. Da der Draht zeigen soll, an welcher Position das eigentliche Messobjekt angebracht werden muss, wäre seine Lage genau auf der Drehachse ideal. Eine kleine Abweichung davon ist akzeptabel. Wichtig dabei ist, dass bei einer späteren Messung das Messobjekt mit deutlich größerer Dicke als der des Drahtes stets im Aufnahmebereich der Kamera zu sehen ist.

Nach Ablauf der Messung stehen die beiden Ergebnisse für alle anderen Programmteile intern zur Verfügung.

Vorgehen zur Ausrichtung des Drahtes:
Standardmäßig sind in der Motorsteuerung zwei Motorpositionen zur optimalen Einstellung des Messobjektes eingerichtet. Wählt man diese Positionen aus, so kann der Draht bestmöglich an der realen Drehachse ausgerichtet werden.

Die Idee:
Um den Ablauf der Messung verstehen zu können ist es sinnvoll, sich ein Schema der Messapparatur vor Augen zu führen:

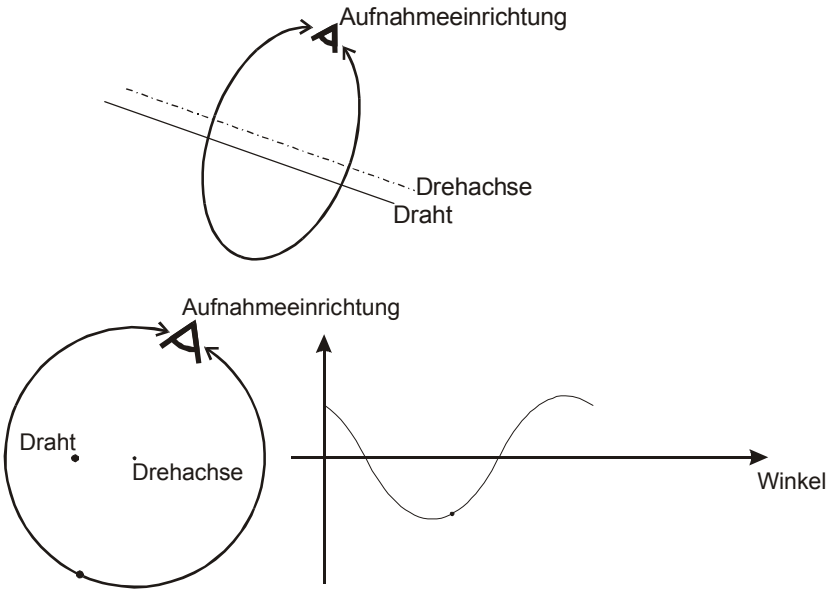
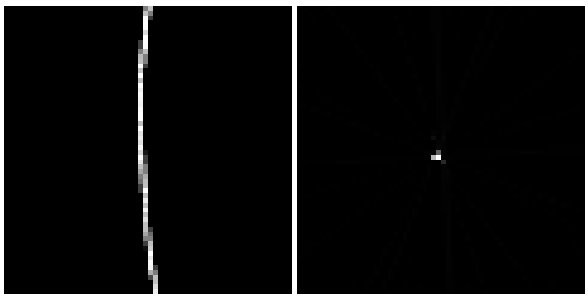


Abbildung 5-13: Schema der Messapparatur

Die Aufnahmeeinrichtung nimmt an diskreten Stellen die Position des Drahtes auf, die dann über dem dazu gehörigen Winkel aufgetragen wird. Es ergibt sich eine ideale Sinuskurve. Aus dieser Kurve können Lage der Drehachse und maximale Amplitude abgelesen werden.

Überprüfung der Genauigkeit und Wichtigkeit der Daten der Drehachsenbestimmung:
Um die Wichtigkeit einer genauen Drehachsenbestimmung zu verstehen, muss man die weitere Verwendung dieser Einstellung im Verlauf einer tomografischen Messung betrachten:
Im Realen bewegt sich die Messapparatur auf einer Kreisbahn um die Drehachse und nimmt an diskreten Stellen Bilder des Messobjektes auf. Diese Bilder müssen in geeigneter Weise beschnitten werden, so dass sich die Drehachse exakt in der Mitte befindet. Dies ist erforderlich, da der Algorithmus dies bei der Berechnung der Flächenfaktoren voraussetzt. (siehe hierzu „Das Formular Simulation“).
Die Auswirkungen, die eine Abweichung der detektierten Drehachse von der realen hat, wurden in den folgenden Messungen bestimmt:
Zunächst einmal die Werte einer Datenaufnahme mit der richtigen Drehachse:



Auf der linken Seite befinden sich die von der Kamera aufgenommenen und anschließend zugeschnittenen Messergebnisse einer punktförmigen Lichtquelle (hier: weiß hinterleuchteter Draht, Bild invertiert). Auf der rechten Seite ist die daraus rekonstruierte Position dieser Lichtquelle zu sehen.

Abbildung 5-14: Punktförmige Lichtquelle mit richtiger Drehachseneinstellung

Um nun zu sehen, wie das rekonstruierte Bild bei falscher Drehachse aussieht, wurden weitere Messungen durchgeführt, bei denen lediglich die Position der Drehachse definiert verändert wurde (durch verändern des Wertes in der Datenbank).

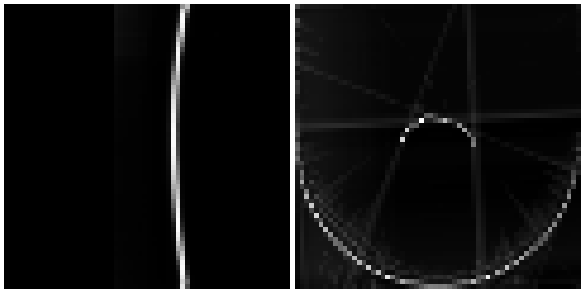


Abbildung 5-15: Punktförmige Lichtquelle mit falscher Drehachseneinstellung

Hier weicht die tatsächliche Drehachse von der detektierten um 48 Pixel bei einer vertikalen Auflösung von 480 ab. Deutlich wird dies zunächst an einer Verschiebung der von der Kamera aufgenommenen Werte um hier 8 Pixel. Lässt man daraufhin den Tomografie-Algorithmus auswerten, so zeigt sich diese Verschiebung in einer Verzerrung der punktförmigen in eine halbkreisförmige Lichtquelle mit Radius 8 Pixel.

Wie kommt nun die Verschiebung um 8 Pixel zustande?
Die vertikale Auflösung der Kamera beträgt 480 Pixel. Zu obiger Messung, die mit einer Auflösung von 60 durchgeführt werden sollte, wurden je 6 Pixel zusammengefasst. Die genutzte vertikale Auflösung betrug also 360 Pixel. Zur ersten Messung befand sich die Drehachse genau in der Mitte, zur zweiten um 48 Pixel verschoben. Relativ zur Auflösung ist dies eine Verschiebung um 13,3%. Betrachtet man also die Auflösung der Auswertung, so setzt sich dieser Fehler hier mit folglich 8 Pixeln fort.

Umgekehrt gefolgert hat man so eine Möglichkeit, die Drehachse per Hand sehr fein zu justieren. Man benötigt lediglich den weiß hinterleuchteten Draht. Ergibt sich nach einer tomografischen Auswertung ein halbkreisförmiges Gebilde so kann man aus dessen Radius mit der Formel:

$$\text{tatsächliche Drehachse} = \text{detektierte Drehachse} \pm \text{zusammengefasste Zeilen} \cdot \text{Radius Halbkreis}$$
$$- \text{ bei Sichelform} \cup; + \text{ bei Sichelform} \cap$$

nachbessern.

Erklärung des Effekts der „Sichelbildung“ bei falsch eingestellter Drehachse

Angenommen der Draht befindet sich exakt in der Drehachse, er wandert also nicht bei einer Drehbewegung auf dem Bildschirm.
Im Fall der richtig eingestellten Drehachse ergibt sich dann ein gerader Strich in der Mitte des Bildes, das aus den aufgenommenen Daten erstellt wurde. Schließlich befindet sich der Draht bei jeder Winkelposition exakt auf der Drehachse, also in der Mitte des Bildes:

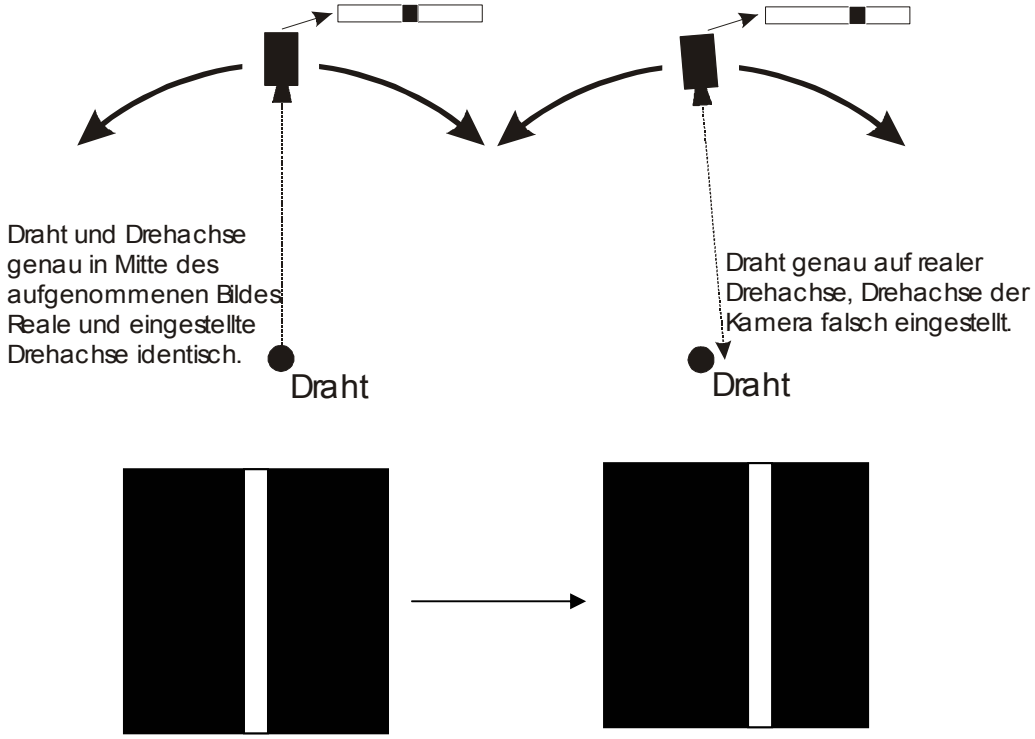

Draht und Drehachse genau in Mitte des aufgenommenen Bildes. Reale und eingestellte Drehachse identisch.
Draht
Draht genau auf realer Drehachse, Drehachse der Kamera falsch eingestellt.
Draht

Abbildung 5-16: Verdeutlichung der Verschiebung der aufgenommenen Daten bei falscher Drehachse

Ist jedoch die Drehachse falsch eingestellt, so ergibt sich ebenfalls ein gerader Strich, jedoch verschoben. Nach wie vor erscheint der Draht bei jeder Winkelposition an der selben Stelle. Da die Kamera jedoch aufgrund der falsch eingestellten Drehachse seitlich darauf blickt, scheint der Draht verschoben.
Genau diese Verschiebung führt zu dem oben genannten Fehler. Betrachtet man die Vorgehensweise des Tomografie-Algorithmus, so wird dies deutlich:
Im für die Erklärung konstruierten Beispiel liegt eine Auflösung von 11 vor, der Tomografie-Algorithmus betrachtet also 11 diskrete Winkelpositionen.

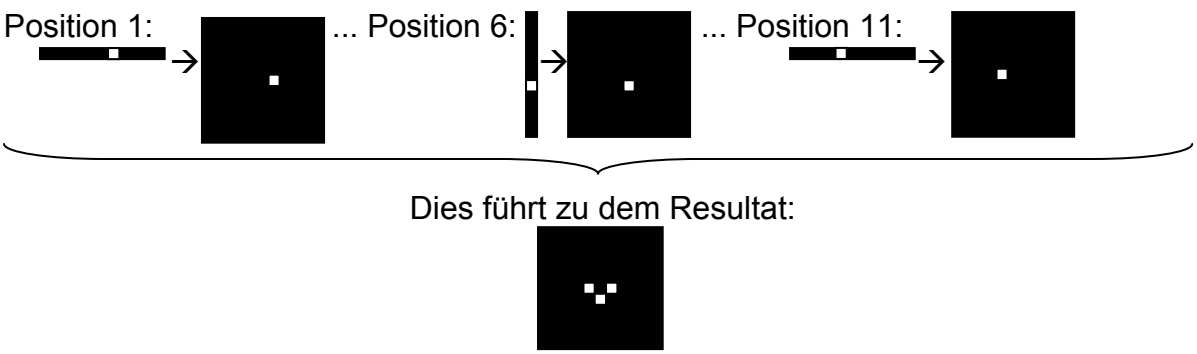


Abbildung 5-17: Zustandekommen der "Sichel" bei falscher Drehachse

Realisierung im Quelltext:

Der Quellcode der **unit** `AutoJustierung` kann in drei Teile aufgeteilt werden:

- MotorThread zur Berechnung der Motorpositionen und Ansteuerung der Schrittmotoren
- Prozeduren zur Auswertung der LiveBilder und Ermittlung der Messergebnisse
- Prozeduren zur Reaktion auf Ereignisse

Nach Drücken des START - Knopfes wird zunächst die Aufnahmeeinrichtung in Ausgangsposition gefahren (`procedure StartJustage()`).

In `TMotorThread.Execute()` wird so lange gewartet, bis die Schrittmotorsteuerung die Ankunft meldet. Nun wird `BildHolen()` aufgerufen, die das aktuelle Standbild der Kamera aufzeichnet und ihrerseits `BildAuswerten()` aufruft.

Dort werden alle Grauwerte einer Zeile aufsummiert und der Pixelwert der Zeile mit der geringsten Grauwertsumme ermittelt. Dieser Wert wird zur weiteren Bearbeitung in `eListe` (TList) gespeichert.

Unterdessen setzt sich MotorThread selbständig auf „suspended“. Mit `MotorThread.Resume` wird jetzt der MotorThread wieder „aufgeweckt“, der zur Aufnahme weiterer Bilder die nächste Position berechnet und anfahren lässt.

Dieser Ablauf wiederholt sich so lange, bis die Endposition (180° gedreht zur Ausgangsposition) erreicht ist.

Es erfolgt die Auswertung der `eListe` in `procedure interpolOnCanvas()`. Die Drehachse wird mit dem ersten und letzten Wert der `eListe` errechnet. Mit Hilfe der Messapparatur kann lediglich ein Bereich von 0° bis 180° abgedeckt werden. Da die aufgenommene Kurve eine Sinuskurve sein muss haben der erste und letzte (durch Punkt gekennzeichnete) Messwert den selben Abstand zur Drehachse. Dies kann man an obiger Skizze ersehen. Im Umkehrschluss muss also die Drehachse genau zwischen den beiden Punkten zum Liegen kommen:

```
axis:=(e1.W+e2.W)/2
```

Der Abstand des Drahtes zur Drehachse ergibt sich dann aus der maximalen Amplitude der Messwertkurve.

Insofern die CheckBox zur Extrapolation der Messwerte ausgewählt wurde, wird nun noch die Kurve bis 360° ergänzt.

5.4.5 Das Formular „Bildbereich auswählen“

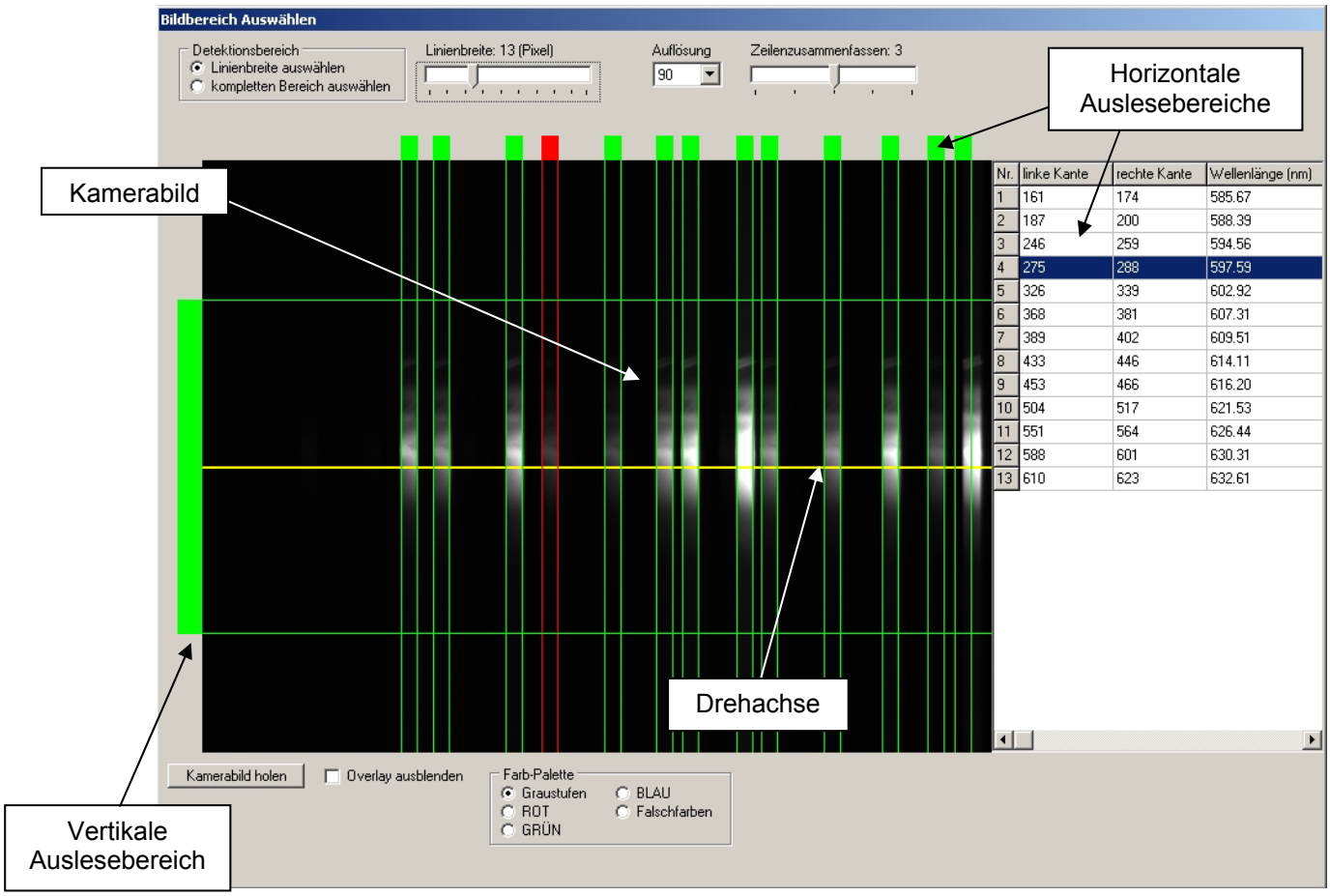


Abbildung 5-18: Das Formular Bildbereich Auswählen

In diesem Formular werden die aufzunehmenden Spektrallinien festgelegt. Es wird hier ebenfalls die Auflösung, die horizontale Abgrenzung der einzelnen Spektrallinien und die vertikale Abgrenzung des gesamten Auslesebereiches festgelegt.
Nachdem mit der Schaltfläche „Kamerabild holen“ ein Livebild aufgenommen wurde, wird automatisch die im Formular „Drehachsen-Bestimmung“ bestimmte Drehachse eingezeichnet. Symmetrisch dazu liegt der vertikal begrenzte Auslesebereich. Die Höhe des Bereiches wird direkt bestimmt durch das Produkt aus Auflösung und Anzahl der zusammengefassten Zeilen. Da der Bildbereich nur maximal 480 Pixel hoch ist, bestimmt die Auflösung gleichzeitig die Anzahl der maximal zusammenfassbaren Zeilen. Ist die Drehachse nicht genau in der Mitte des Bildes, werden durch das Programm Auslesebereiche, die über die Bildgrenze hinausgehen würden, verboten.
Über die `RadioGroup` Detektionsbereich kann neben dem normalen Messmodus eine Art Formmessung (ohne spektrale Auflösung) der Quelle durchgeführt werden. Durch Auswahl des kompletten Bereiches werden keine einzelnen Spektrallinien aufgenommen, sondern sämtliche Emissionen des Messobjektes im Aufnahmebereich der Kamera. Zu beachten ist dabei allerdings, das auf Grund der begrenzten spektralen Auflösung des Messaufbaus nicht die gesamte Emission des Messobjektes aufgenommen wird.
Wird jedoch die Linienbreite festgelegt, befindet sich das Messprogramm im Normalmodus.
Durch Rechtsklick auf das aufgenommene Bild können jetzt beliebig viele horizontal begrenzte Auslesebereiche hinzugefügt werden. Durch Verschiebung mit der Maus

lässt sich ihre jeweilige Position festlegen. Die Linienbreite (eine konstante Größe, die vom Spektrografen abhängt) für alle Linien wird oben links auf diesem Formular festgelegt. Eine Tabelle am rechten Formularrand gibt gleichzeitig die linke und rechte Grenze der einzelnen Bereiche an. Wurde für das Spektrum bereits eine spektrale Justage durchgeführt, kann in dieser Tabelle ebenfalls die Mittenwellenlänge des jeweiligen Bereiches abgelesen werden. Ansonsten ist diese Spalte leer. Durch Rechtsklick auf die Tabelle, kann der angeklickte Bereich gelöscht werden.
Wie auch in einigen anderen Formularen, können im unteren Bereich verschiedene Falschfarbendarstellungen gewählt werden bzw. es kann die gesamte Anzeige der Bereiche abgeschaltet werden. Eine Verschiebung der Bereiche ist allerdings nur möglich, wenn diese auch angezeigt werden.

Das Formular übernimmt die im Hauptformular eingestellten Werte für Bildinvertierung und Farbbereichsanpassung.

Die Darstellung der horizontalen Bereiche wurde durch die dynamische Erstellung von `Panels` realisiert, die in einer eigenen Klasse `TMyCustomPanel` zusammengefasst sind und alle in einer Liste gespeichert werden. Die Klasse enthält genau drei `Panels`: eines für die jeweils linke und rechte Grenze, sowie ein `Panel` für den Kopfbereich.

```
TMyCustomPanel =  Class
                                Panel1:TPanel;
                                Panel2:TPanel;
                                Panel3:TPanel;
                                NP:TPoint;
                                OS: Integer;
                                Orientation: GOrientation;
                                Constructor Create(Owner: TWinControl; Nullpkt: TPoint;
                                            Offset: Integer;
                                            Const POrientation: GOrientation = aVertical;
                                            Const Visible: Boolean = True);
                                Destructor Destroy();
                                function getDaten:TStrings;
                                procedure BreiteSetzen(Offset: Integer);
                                procedure BringToFront();
                                procedure verschieben(X: Integer);
                                function Sort(Item1, Item2: Pointer): Integer;
                                procedure Hide();
                                procedure Show();
                                procedure MoveTo(X: Integer);
                           end;
```

Im `Constructor` werden gleich bei der Erstellung alle notwendigen Merkmale übergeben z.B. Orientierung (horizontal oder vertikal), die Position und das Hauptfenster in dem sie dargestellt werden sollen.
Die einzelnen Methoden dieser Klasse erlauben ein Verschieben, Verändern der Breite, Sortieren der Panels und Auslesen der Paneleigenschaften.
Sollen sich Panels mit der Maus bewegen lassen, so muss bei ihrer Erstellung wie im folgenden Codeabschnitt beschrieben eine dementsprechende Mouse – Funktion geschrieben und zugewiesen werden:

```
var
        tmp:TMyCustomPanel;
begin
…
        tmp.Panel3.OnMouseDown:=PanelMouseDown;
        tmp.Panel3.OnMouseUp:=PanelMouseUp;
        tmp.Panel3.OnMouseMove:=PanelMouseMove;
…
end;
```

Diese Funktionen bestimmen dann das Verhalten der einzelnen Panels. In unserem Fall erlauben sie eine Bewegung auf dem Kamerabild entweder in horizontaler oder vertikaler Richtung bis zur Bildgrenze.

5.4.6 Das Formular „Messung“

Das Formular Messung bietet eine Zusammenfassung aller für die Messung benötigten Werte.
Mit den beiden Schaltflächen „Laden“ und „Speichern“ bietet es ebenfalls die Möglichkeit die Daten einer Messung abzuspeichern bzw. zu laden. Beim Laden der Daten werden dann alle Werte des Programms durch diese ersetzt.
Da auch messtechnisch ermittelte Werte wie „Zeile der Drehachse“ oder gemessene Wellenlängen geladen werden, muss vom Benutzer sichergestellt werden, dass seit der Speicherung diese nicht verstellt / verändert wurden, da sonst eine sinnvolle Messung nicht mehr möglich ist. Die Schaltfläche „Messung starten“ dient zum Start der Messung. Ein Flussdiagramm zum Messablauf ist in Abbildung 5-21 zu sehen.

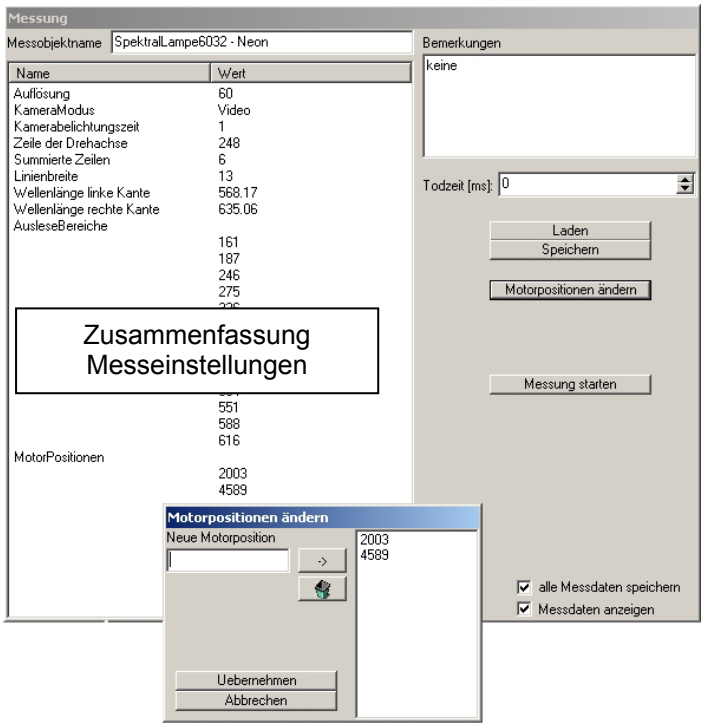


Abbildung 5-19: Das Formular Messung

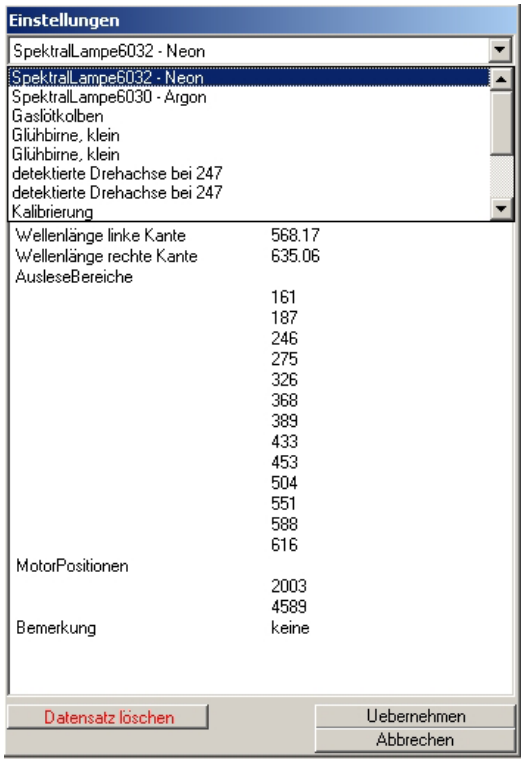


Abbildung 5-20: Unterformular Einstellung

Das Formular stellt eine zentrale Sammelstelle für alle im Programm wichtigen Zustände und Daten dar. Dort kann vor der Messung eine letzte Kontrolle stattfinden.
Mit der `Checkbox` „alle Messdaten speichern“ wird das Programm veranlasst alle aufgenommenen Bilder für eine spätere Auswertung abzuspeichern. Mit dem Feld Totzeit wird die Zeit festgelegt, die nach Fahrbewegung gewartet wird bis ein Bild aufgenommen wird. Es zeigte sich allerdings, dass sich die Aufnahmen nicht verändern. Eine Beeinflussung der Aufnahmen durch eventuelles Nachwippen / Nachwackeln der Apparatur nach Abschluss der Fahrbewegung kann also ausgeschlossen werden.
Mit der Schaltfläche „Motorpositionen ändern“ kann ein kleines Dialogfeld geöffnet werden, in dem die gewünschten Motorpositionen (Fahrbewegung) eingegeben werden können.
Das Unterformular Einstellungen ist rechts zu

sehen. Es besteht im Wesentlichen nur aus einer `Combobox`, aus der die Datensätze nach Messobjekt ausgewählt werden können und einer Listenansicht, die eine Sichtprüfung der gespeicherten Werte anzeigt. Das Formular selbst wird modal angezeigt. Es sperrt somit die komplette Anwendung bis es wieder geschlossen wird. Durch Wählen der Schaltfläche „Abbrechen“ werden keine Änderungen im Programm vorgenommen, „Datensatz löschen“ löscht unwiderruflich den aktuellen Datensatz, „Uebernehmen“ liest die gewählten Einstellungen in das Programm ein und ersetzt eventuell vorhandene Werte unwiderruflich.

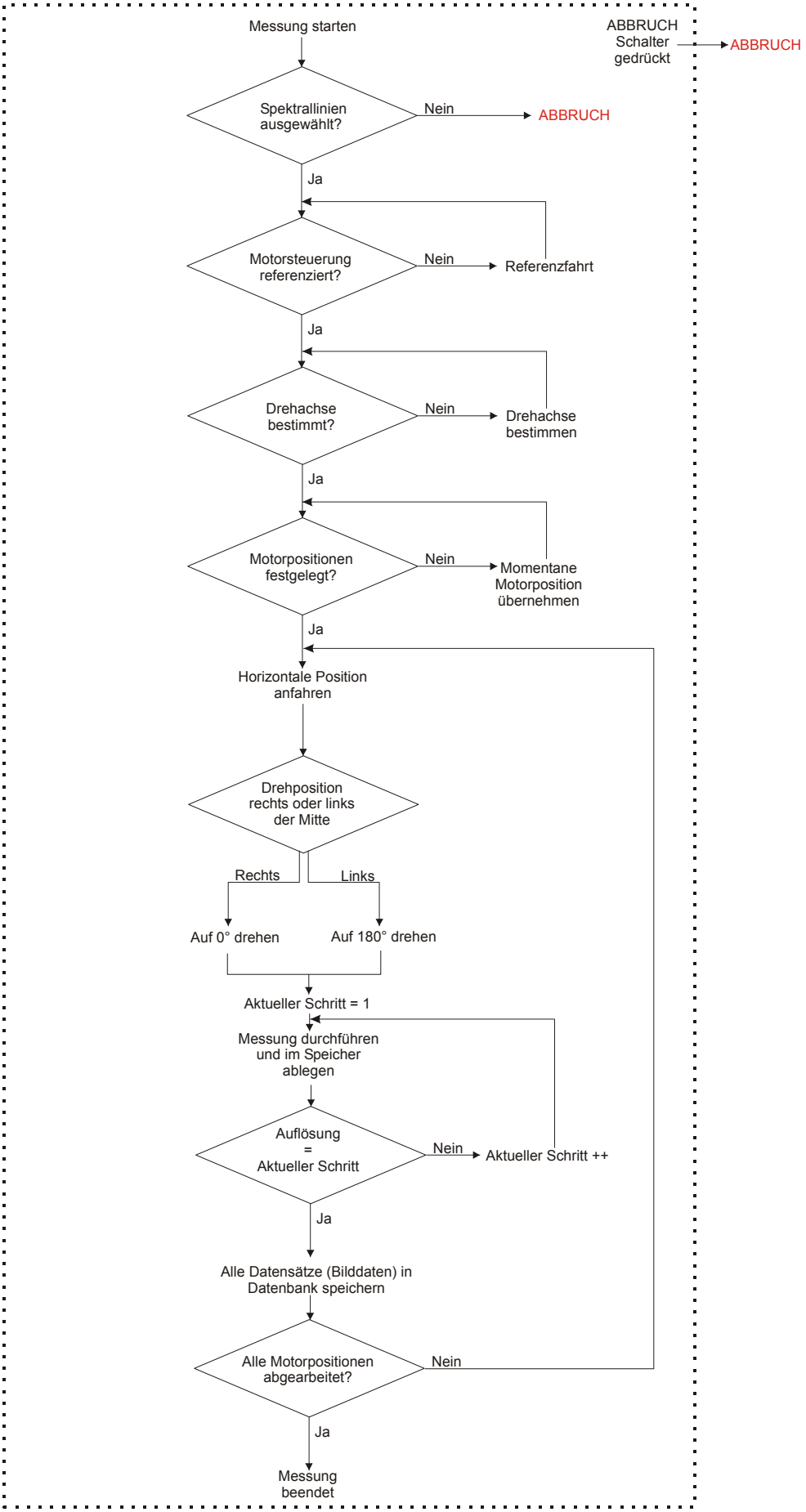


Abbildung 5-21: Ablaufdiagramm Messablauf

Um eine Messung durchführen zu können, müssen vorher in den vorangegangenen Formularen bestimmte Einstellungen vorgenommen worden sein. So muss die Steuerung eingeschaltet und die aktuelle Position durch eine Referenzfahrt festgestellt werden. Ebenfalls notwendig ist, dass eine Drehachse bestimmt wurde. Es müssen ebenfalls Spektrallinien ausgewählt werden, die aufgenommen werden sollen. Diese Einstellungen können geladen bzw. durch das Programm selbst ermittelt sein, damit die Messung beginnen kann.
Stehen diese Einstellungen zur Verfügung, wird geprüft an wie vielen verschiedenen Horizontalpositionen Messungen vorgenommen werden sollen. Dementsprechend viele Positionen werden im Lauf der Messung angefahren. Sollten keine Motorpositionen gewählt worden sein, wird nur an der aktuellen Position eine Messung durchgeführt.
Das Programm stellt nun - um Zeit zu sparen - den kürzesten Weg zu einer der Startpositionen fest und führt dann die Messung gegen oder mit dem Uhrzeigersinn durch. Um sie später vergleichen zu können werden die aufgenommenen Bilder jedoch so sortiert, als handle es sich um eine Messung gegen den Uhrzeigersinn.
Die Anzahl der einzelnen Schritte wird durch die Auflösung bestimmt. So werden zum Beispiel für eine Messung mit der Auflösung 60 bei jeder Drehung genau 180°/60 gedreht und das 60-mal. An jeder Position werden nun alle Spektrallinien gleichzeitig aufgenommen und im Speicher gepuffert, und am Ende der 60 Schritte auf Festplatte gesichert. Ein Screenshot wird jeweils an der Nullposition gespeichert. Falls alle Bilder aufgenommen werden sollen, werden auch diese gespeichert. Das Datenvolumen nimmt dabei mit der Auflösung zu. Die Größe der Datei beträgt dabei ohne Header: Auflösung · 600 KB (min. 18 MB, max. 108 MB) pro Motorposition.
Sind alle Motorpositionen abgearbeitet, ist die Messung beendet.

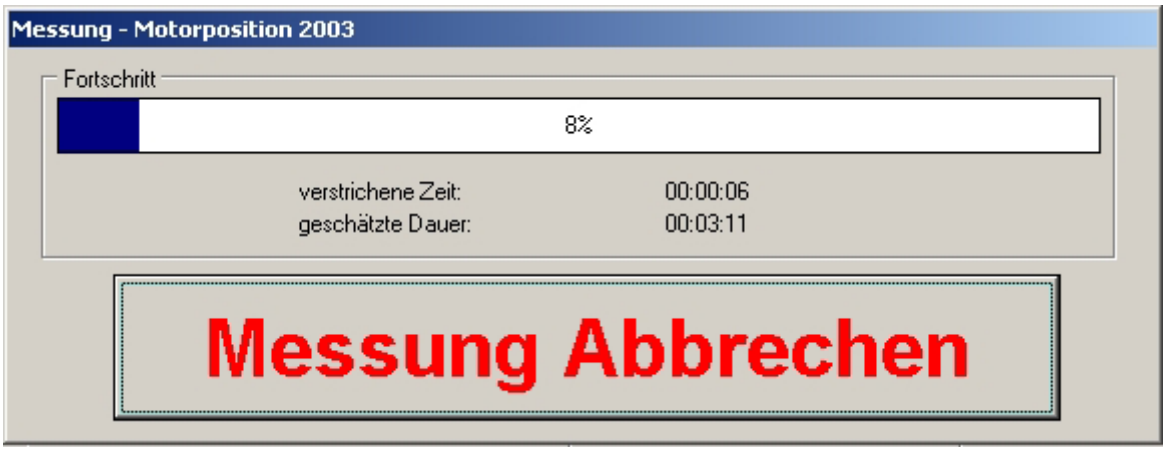


Abbildung 5-22: Anzeige während der Messung

Während der Messung wird das in Abbildung 5-22 dargestellte Formular auf dem Bildschirm angezeigt. Es dient nur dem Zweck die automatische Messung abbrechen zu können. Durch Klicken auf den Schalter „Messung Abbrechen“ wird die Messung zum nächstmöglichen Zeitpunkt beendet. Ist auf dem Formular Messung (Abbildung 5-19) die Checkbox „Messdaten anzeigen“ gewählt, werden unter der Schaltfläche „Messung Abbrechen“ für jede Spektrallinie eine kleine Preview der Messdaten angezeigt.

Außerdem versorgt es den Benutzer mit einigen Statusinformationen wie:

- Verstrichene Zeit seit Messbeginn
- Geschätzte Dauer bis zum Ende der gesamten Messung
- Forschrittsbalken der aktuellen Motorposition
  (blau: normaler Status, rot: Steuerung wartet Totzeit ab)
- Titelleiste mit aktuell angefahrener Motorposition

Während dieses Formular auf der Oberfläche aktiv ist sind keine Eingaben an das Programm möglich.
Bei Messende wird dieses Formular automatisch geschlossen.

Die Messung erfolgt durch einen eigenen Thread. Somit bleibt die komplette Funktionalität der Anwendung erhalten. Aus Sicherheitsgründen werden allerdings während der Messung alle nicht notwendigen Formulare geschlossen bzw. versteckt. Um den Ablauf verfolgen zu können, bleibt einzig das Formular zur Einstellung der Auslesebereiche sichtbar. Dieses wird während der gesamten Messung mit einem Livebild aktualisiert, ist allerdings für Eingaben gesperrt. Der Thread selbst ist wie folgt aufgebaut:

```
TMessThread = class(TThread)
                private
                        AktuellerSchritt: Integer;
                        RefTime, NowTime, VorTime: TDateTime;
                        Daten: array of TMemoryStream;
                        Aufloesung: Integer;
                protected
                        Direction: TDirection;
                        procedure Execute(); override;

                        procedure Vorbereiten();                          ①

                        procedure Referenzieren_vorbereiten();            ②
                        procedure Referenzieren_abbrechen();

                        procedure Justage_Vorbereiten();                  ③
                        procedure Justage_Abbrechen();
                        Procedure Justage_nachbereiten();

                        procedure Messung_vorbereiten();
                        procedure Messung_Fahren();                       ④
                        procedure Messung_Durchfuehren();
                        procedure Messung_abbrechen();

                        procedure Time_Update();

                        procedure Aufraeumen();                           ⑤
                public
                        constructor Create(Suspended: Boolean);
                end;
```

Zu Beginn einer jeden Messung wird der Thread mit seinem `Constructor` erstellt:

```
constructor TMessThread.Create(Suspended: Boolean);
begin
        inherited Create(Suspended);
        Priority:=tpHigher;
end;
```

Die Priorität des MessThread wird dabei auf die Stufe Hoch festgelegt. So wird die Ausführung der Messung vor allen anderen Aktivitäten der Anwendung bzw. des Systems sichergestellt. Die Messung muss so kurz wie möglich sein, um eine unnötige Aufheizung des Messaufbaus zu verhindern. Auf diese Weise werden Zeitverzögerungen durch Verlust des Rechnerkerns für nicht notwendige Vorgänge minimiert.

Die Procedure `Execute()` übernimmt die Steuerung des Threads. Dieser ist in fünf große Blöcke eingeteilt (siehe Klassendefinition):

1. Vorbereitungen zur Messung
2. Referenzfahrten
3. Justage
4. Messung
5. Nachbereitung der Messung

Alle diese Teilbereiche nehmen Änderungen an der Oberfläche des Hauptprogramms vor. Sie werden deshalb aus Sicherheitsgründen nur mit „Synchronized“ aufgerufen. Greifen z.B. die Hauptanwendung gleichzeitig der Thread auf die Oberfläche zu, wäre dieser Aufruf nicht „Synchronized“. Es käme es zu einem Systemabsturz, da nicht zwei Threads gleichzeitig ein Objekt ändern können.
Zur Vorbereitung der Messung werden alle offenen Fenster, in denen Einstellungen vorgenommen werden könnten, geschlossen. Sind keine spektralen Auslesebereiche festgelegt wird die Messung abgebrochen. Das in Abbildung 5-18 dargestellte Formular wird initialisiert. Es wird nun überprüft ob nicht sinnvolle Einstellungen vorgenommen wurden. So macht es keinen Sinn eine Spektrallinie mehrmals in die Datenbank zu übernehmen. Die Daten wären identisch. Deshalb werden doppelte Messungen entfernt.
Um einen möglichst automatisierten Messablauf durchzuführen kann der Thread jetzt im zweiten Schritt feststellen, ob die Steuerung schon referenziert wurde. Ist dies nicht der Fall, führt er automatisch eine Referenzfahrt durch.
Auf ähnliche Weise geschieht dies im dritten Teil der Justage der Drehachse. Ist diese noch nicht festgelegt wird der Bediener aufgefordert den Messaufbau herzustellen damit eine Feststellung möglich ist.
Nun kann mit dem vierten und Hauptteil begonnen werden. Dazu muss erst überprüft werden, welche horizontalen Motorpositionen angefahren werden sollen. Sind hier keine vorgeben, wird die aktuelle Position als einzige gemessen. Sind allerdings mehrere definiert, werden sie der Reihe nach angefahren.
Die Procedure die dann die Drehbewegung übernimmt sieht wie folgt aus:

```
procedure TMessThread.Messung_Fahren();
begin
        if Direction=CCW then begin
                if AktuellerSchritt=0 then
                        MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-
                                1)*(AktuellerSchritt)+StartAngle), 5000)
                else begin
                        if Aufloesung>90 then
                                MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-
                                1)*(AktuellerSchritt)+StartAngle), 2000)
                        else
                                MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-
                                1)*(AktuellerSchritt)+StartAngle), 4000);
                end;
        end
        else if Direction=CW then begin
                if AktuellerSchritt=0 then
                        MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-1)*(Aufloesung-
                                AktuellerSchritt-1)+StartAngle), 5000)
                else begin
                        if Aufloesung>90 then
                                MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-
                                1)*(Aufloesung-AktuellerSchritt-1)+StartAngle), 2000)
                        else
                                MotStForm.DrehenNach(Round(WorkAngle/(Aufloesung-
                                1)*(Aufloesung-AktuellerSchritt-1)+StartAngle), 4000);
                end;
        end;
end;
```

Als erstes unterscheidet das Programm ob links oder rechts herum gedreht wird. Anschließend wird beim ersten Schritt mit Höchstgeschwindigkeit zum Ausgangspunkt gefahren. Um die Motoren nicht übermäßig zu beanspruchen wird in allen weiteren Durchläufen je nach Auflösung mit unterschiedlicher Geschwindigkeit gedreht.
Damit das Bild erst aufgenommen wird, wenn der Motor seine Ausgangsposition erreicht hat, bedarf es eines bestimmten Verfahrens. Dieses Verfahren soll nun anhand eines Beispiels erklärt werden. Zunächst der dafür notwendige Delphicode:

```
FahrEvent.ResetEvent;
if not Terminated then Synchronize(Messung_Fahren);        //Anfahren der Positionen
while WaitForSingleObject(FahrEvent.Handle, 1000)<>0 do begin
     if ABBRUCH_Messung then begin
        terminate;
        break;
      end;
      // bei Bedarf weitere Befehle …
end;
```

Der ganze Wartezyklus läuft eventgesteuert ab. Es werden also Botschaften von anderen Programmteilen an den Thread zur Messung geschickt.
Mit dem Befehl `FahrEvent.ResetEvent` wird dafür gesorgt, dass das Event welches die abgeschlossene Fahrt anzeigt, zurückgesetzt wird. Jetzt kann, wie schon erklärt, die Methode zur Fahrt `synchronized` aufgerufen werden.
Die folgende `While` – Schleife ist nun für die Abfrage des Events notwendig. Der Befehl `WaitForSingleObject(<Event>, <Wartezeit>)` hält den Thread genau so lange an bis entweder das Event ausgelöst wird oder die Wartezeit abgelaufen ist. Dabei wird keine Rechenzeit der CPU benötigt. Die Anwendung kann also weiter ohne Störung ausgeführt werden. Um die Messung dennoch abbrechen zu können, wird in der `While` – Schleife zusätzlich eine boolesche Variable abgefragt. Ist die Variable `ABBRUCH_Messung` seit dem letzten Mal durch eine Eingabe auf True gesetzt worden, wird nicht mehr weiter gewartet sondern die Messung sofort abgebrochen und letzte Aufräumarbeiten ausgeführt. Die Schleife wird, wie im Codeausschnitt zu sehen, je Sekunde einmal durchlaufen. Sollte der Motor schon eher fertig sein, wird jedoch schon früher die Programmausführung fortgesetzt.

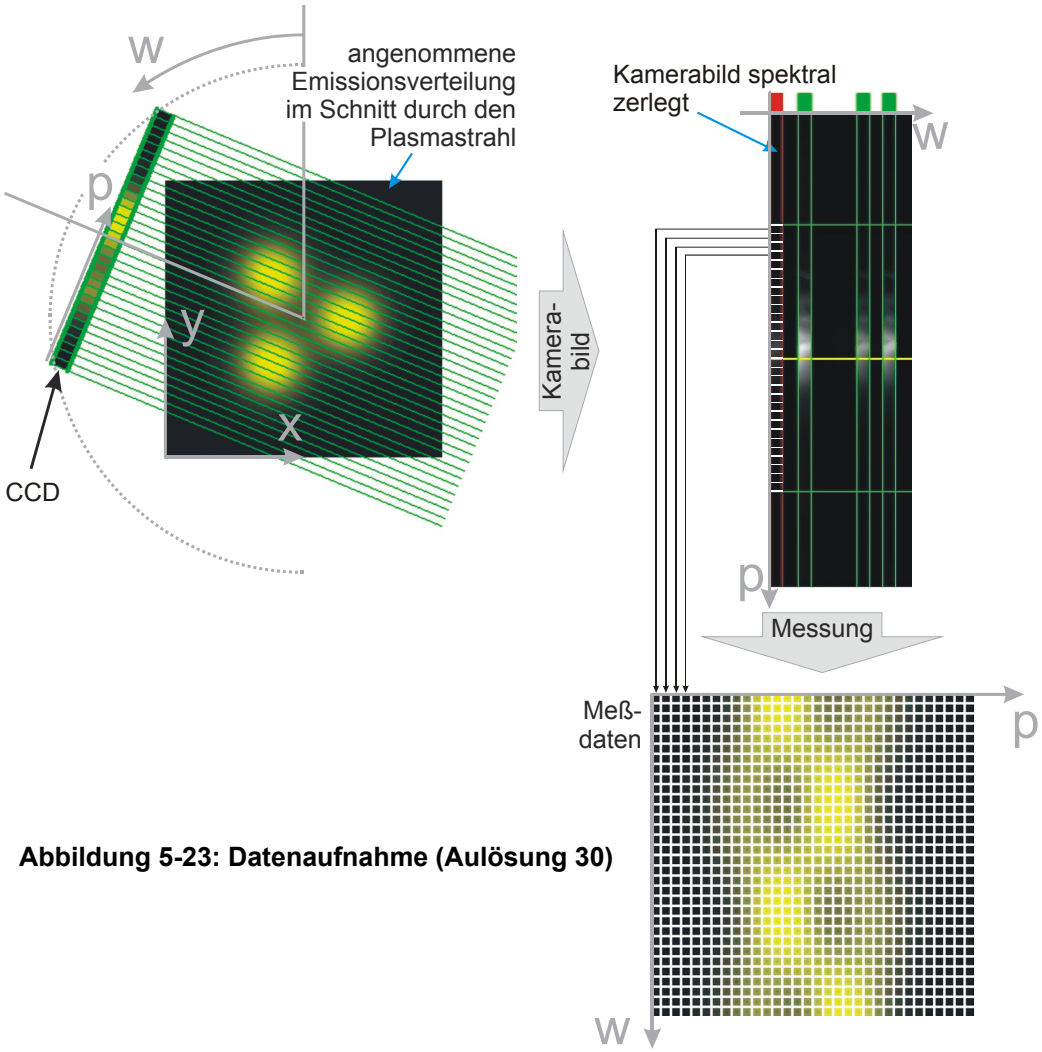


Abbildung 5-23: Datenaufnahme (Aulösung 30)

Zur Datenaufnahme wird jetzt ein Kamerabild aufgenommen und die markierten Bereiche werden ausgelesen.
Dabei werden die einzelnen Spektralauslesebereiche je nach Auflösung zerlegt. Dabei entsteht für jeden Bereich ein Messwert. Dieser 4 – Byte – Wert wird in einem RAM - Stream gespeichert, der später in die Datenbank geschrieben wird. Bei einer Messung ergeben sich demnach genau so viele Streams wie angewählte Messbereiche. Im Beispiel der Abbildung 5-23 sind es vier Streams mit jeweils 30 Werten pro angefahrener Position, also insgesamt 900 4 – Byte – Werte.
Sind diese Messungen alle abgeschlossen und auch alle Motorpositionen ausgewertet, wird im fünften Schritt der Speicher freigeben und die Steuerung wieder freigeschaltet. Anschließend beendet sich der Thread mit `Terminate`. Damit wird verhindert, dass er bei der nächsten Messung undefiniert wieder aufgerufen werden kann. Er muss komplett neu erstellt werden.

Der MessThread selbst hat, wie schon beschrieben, die Möglichkeit alle Bilder abzuspeichern. Hierbei sind noch einige Dinge zu beachten:

- Es entstehen sehr große Datenmengen auf Grund der Kameraauflösung (Auflösung * 640 * 480 * 16 Bit → 18 MB – 108 MB).
- Die Speicherung erfolgt nicht in der Datenbank selbst → Bilddateien müssen Zusatzdaten enthalten.

Der Aufbau der Bilddateien ist trotzdem einfach. In ihr werden alle aufgenommenen Bilder einer Motorposition sequenziell gespeichert. Sie besteht somit immer aus 600 KB großen Datenblöcken, die ein Bild enthalten. Je nach Auflösung entstehen Datei-

en mit 30 bis 180 Bildern. Diese werden auf Grund der Geschwindigkeit als ganze Blöcke gespeichert:

```
Var
        f: File of Word;
…
        try
                    AssignFile(f, <DATEINAME DATUMSKODIERT>.TomoSerien);
                    ReWrite(f);

                    BlockWrite(f, BildDaten12Bit^, 640*480); // Wiederholung je nach Auflösung


                    CloseFile(f);
        except
        end;
```

Nachdem mit `AssignFile` und `ReWrite` eine Datei erstellt wurde, können nun mit `BlockWrite` die gesamten Bilddaten auf einen Schlag auf die Festplatte geschrieben werden. Dementsprechend kann man sie später mit `BlockRead` auf die gleiche Weise laden. Der Dateiname ist ähnlich den abgespeicherten Spektrallinien wie folgt kodiert:

- Aufnahmezeitpunkt im julianischen Datumsformat
- Motorsposition der Aufnahme
- Dateiendung „TomoSerien“

Sind alle Bilder gespeichert, wird die Datei vom Typ „`File of Word`“ wieder geschlossen.
Um später eine Zuordnung der Datei zu einer Messung bzw. Messreihe zu ermöglichen, wurde an die Datei selbst eine „Header“ angehängt. Angehängt deshalb, damit es weiterhin möglich ist, dass Formulare, die nur TomoBilder lesen können ebenfalls in der Lage sind, das erste Bild der TomoSerien – Dateien anzuzeigen.
Um eine aufwendige Speicherung von Binärdaten zu umgehen, wurde der schon beschriebene Record `Header` verwendet. Da es sich bei der Datei nicht um eine typisierte Datei dieses Typs handelt, muss dazu allerdings ein kleiner Trick angewendet werden:

```
Var
        FileHeaderFile: File of Header;
…
                AssignFile(FileHeaderFile, <DATEINAME DATUMSKODIERT>.TomoSerien);
                ReSet(FileHeaderFile);
                Seek(FileHeaderFile, FileSize(save)+1);                 //+1: leere Daten
                                //anhängen bis Blockgröße erreicht, dann Header 'ankleben'
                Write(FileHeaderFile, FileHeader);
                CloseFile(FileHeaderFile);
```

Dazu wird dieselbe Datei nachdem sie geschlossen wurde erneut geöffnet. Diesmal allerdings als Datei vom Typ „`File of Header`“. Jetzt kann mit `Seek` und `Filesize` das Ende der Datei gesucht werden. Delphi kontrolliert dabei nicht den Inhalt der Datei. Es springt lediglich zum Ende der Datei an die Stelle, an der der letzte ganze Datensatz beginnen würde. Dies ist aber bei uns nicht der Fall, da unsere abgespeicherten Bilder eine andere Größe haben als die Datenstruktur `Header`. Es sind deshalb immer noch Bilddaten nach dieser Sprungmarke gespeichert. Allerdings weniger als ein kompletter Datensatz groß wäre. Deshalb springen wir zum Ende der Datei und gehen noch einen Datensatz weiter. Delphi hängt hier leere Bytes am Ende der Datei an. Jetzt können wir ohne unsere Bilddaten zu vernichten den von uns gewünschten `Header` mit `Write` anhängen.

Eine hexadezimale Anzeige des Dateiinhaltes verdeutlicht das Prinzip:

Reine Bilddaten:

Zeile	Hexadezimal dargestellter Inhalt
…	Bilddaten
06977FF0	04 01 10 01 09 01 13 01 10 01 11 01 08 01 13 01

Bilddaten und Header:

Zeile	Hexadezimal dargestellter Inhalt
…	Bilddaten
06977FF0	04 01 10 01 09 01 13 01 10 01 11 01 08 01 13 01
06978000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
…	Leerer Pufferplatz
069781B0	00 00 00 00 00 00 00 00 48 CE 74 9A 71 60 E2 40
…	weitere Daten
069783E0	40 0B 8C 00 79 3C 42 00 81 3C 42 00 90 F3 12 00

Die Größenzunahme laut Dateibrowser beträgt hier 1,008 KB. Die Vergrößerung der Datei von nur einem KB liegt somit noch innerhalb der Zuordnungseinheitengröße von Windows und somit in einem Bereich, der bei nahezu allen Windowssystemen sowieso nicht benutzbar wäre.
Das Laden der Headerinformation geschieht auf die gleiche Weise wie das Speichern. Öffnen der Datei, Dateizeiger auf den letzten Datensatz setzen und Lesen in eine leere Variable vom Type `Header`.

5.4.7 Das Formular „Spektrale-Justage“

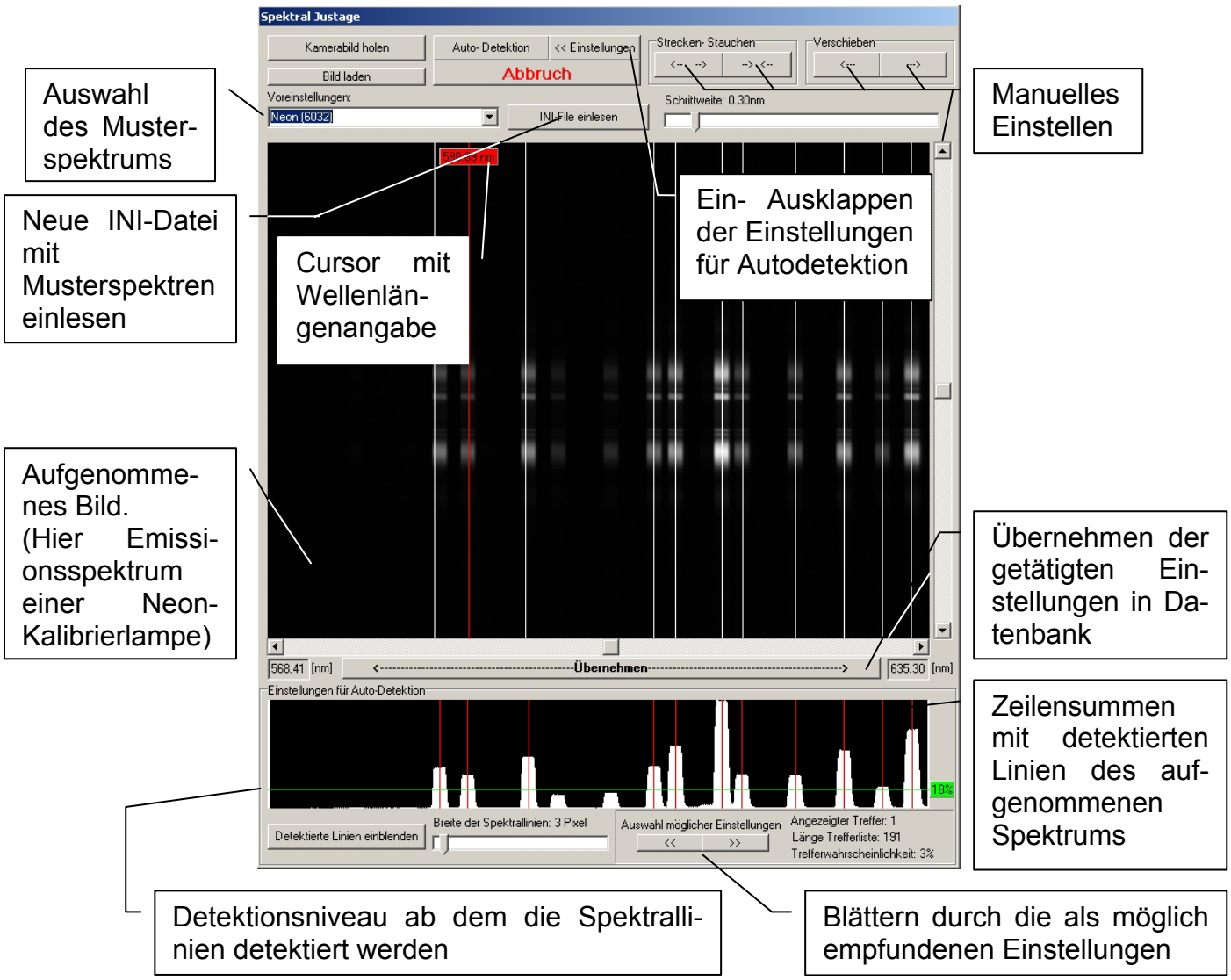


Abbildung 5-24: Das Formular "Spektral-Justage"

Beschreibung:
Mit der Messapparatur sollen spektroskopische Messungen durchgeführt werden. Leider liefert uns die Kamera keine Aussagen über die aufgenommenen Wellenlängen. Das einzige, was wir wissen, ist, dass der der Kamera vorgeschaltete Spektrograf ein nach Wellenlängen sortiertes, näherungsweise lineares Spektrum bereitstellt. Um nun Aussagen darüber machen zu können, welche Wellenlänge wo auf dem Bild zu finden ist, wurde dieses Formular entwickelt.

Automatische Justierung:
Die Automatische Justierung ist so ausgelegt, dass sie in den meisten Fällen ein richtiges Ergebnis liefert. Trotz allem kann es vorkommen, dass der Benutzer selbst Einstellungen vornehmen muss. Worauf hierbei zu achten ist, wird im Weiteren beschrieben.

Zunächst einmal zum **idealtypischen Ablauf**:
Das zu untersuchende spektral aufgelöste Bild kann auf zweierlei Arten geladen werden: Direkt von der Kamera oder als gespeichertes Bild (TomoBild oder 24BitBitmap).

Voraussetzung für eine realistische Justierung ist:

- Das zu untersuchende Spektrum muss mit seinen charakteristischen Linien in der INI-Datei vorhanden sein (zur Bearbeitung der INI- Datei siehe weiter unten)
- Die Spektrallinien im Bild sollten möglichst schmal und scharf abgebildet sein (Diracförmig)
- Das Bild sollte an keiner Stelle überbelichtet sein (siehe hierzu das Formular „Aufnahme“), also nirgends den maximalen Grauwert aufweisen.
- Es sollten mindestens vier Spektrallinien vorhanden sein.

Nachdem das Bild zu sehen ist, kann nun am DropDown-Menü Voreinstellungen das Musterspektrum ausgewählt werden: Die Musterspektrallinien werden sichtbar.
Theoretisch kann bereits jetzt mit der Auto-Justage begonnen werden, hierzu bitte Button Auto-Justage betätigen. Da der Justage- Algorithmus sehr rechenintensiv ist kann es vorkommen, dass der Computer für kurze Zeit nicht reagiert.
Nachdem der Algorithmus zu Ende gelaufen ist, wird sofort die als am wahrscheinlichsten erachtete Einstellung angezeigt. Liegen nun möglichst viele Musterlinien auf den Spektrallinien des Bildes, jedoch nur jeweils eine pro Bild-Linie, so ist die Einstellung beendet und die Werte der Wellenlängen an der linken und rechten Bildkante können in die Datenbank übernommen werden.
Ist dies nicht der Fall, so sind nun **Einstellungen von Hand nötig**!

Fall 1: Die automatisch erhaltenen Einstellungen sind sinnvoll, könnten aber noch etwas **verschoben** werden, um noch bessere Deckung zu bekommen.
Um die Musterlinien nach links oder rechts zu verschieben, steht zum Einen eine ScrollBar am unteren Bildrand oder die Buttons <-- und --> zur Verfügung. Zusätzlich kann eine Streckung oder Stauchung durch die ScrollBar am rechten Bildrand oder durch die Buttons <-- --> und --><-- erreicht werden.

Fall 2: Das Ergebnis ist überhaupt nicht zufriedenstellend.
Der erste zu empfehlende manuelle Eingriff wäre hier das Auswählen einer als weniger wahrscheinlich erachteten Einstellung. Da der Algorithmus lediglich Abstandsverhältnisse zwischen den Spektrallinien betrachtet, so kann es sein, dass eigentlich unmögliche Einstellungen als am wahrscheinlichsten detektiert werden. Oft verbirgt sich dann die richtige Lösung unter den ersten zehn der Trefferliste.
Auswählbar ist dies nach „Ausklappen“ der Einstellungen >> an den Buttons << und >>.

Führt dies auch nicht zum gewünschten Ergebnis, so stehen noch zwei weitere Einstellmöglichkeiten zur Verfügung:

- Variieren des *Detektionsniveaus*: Ideal ist es, wenn alle im Bild zu sehenden Spektrallinien auch vom Algorithmus detektiert werden; also Detektionsniveau möglichst niedrig legen - allerdings nicht so niedrig, dass mehrere Linien als eine detektiert werden. Überprüfbar ist dies mit dem Button detektierte Linien einblenden.

- *Breite der Spektrallinien*: Eine Verringerung bewirkt eine Verkleinerung der Toleranz, was wiederum die Länge der Trefferliste begrenzt. Dies ist bis zu einem gewissen Punkt durchaus sinnvoll, zu klein sollte sie jedoch nicht gewählt werden, da sonst auch die richtige Lösung als unwahrscheinlich verworfen wird.

  Zweckmäßig ist die Einstellung auf die Breite der vorhandenen Spektrallinien.

Während der Veränderung dieser Daten sollte man regelmäßig die Taste
Auto- Justage betätigen; die Wahrscheinlichkeit ist groß, dass nun die richtige Einstellung detektiert wird!

Abschließend sei noch zu bemerken: Sind mögliche Werte für linke und rechte Grenze bereits bekannt, so können sie auch direkt an den beiden Edit-Feldern links und rechts unter dem Bildrand eingegeben werden!

Die verschiedenen Einstellmöglichkeiten zusammengefasst:

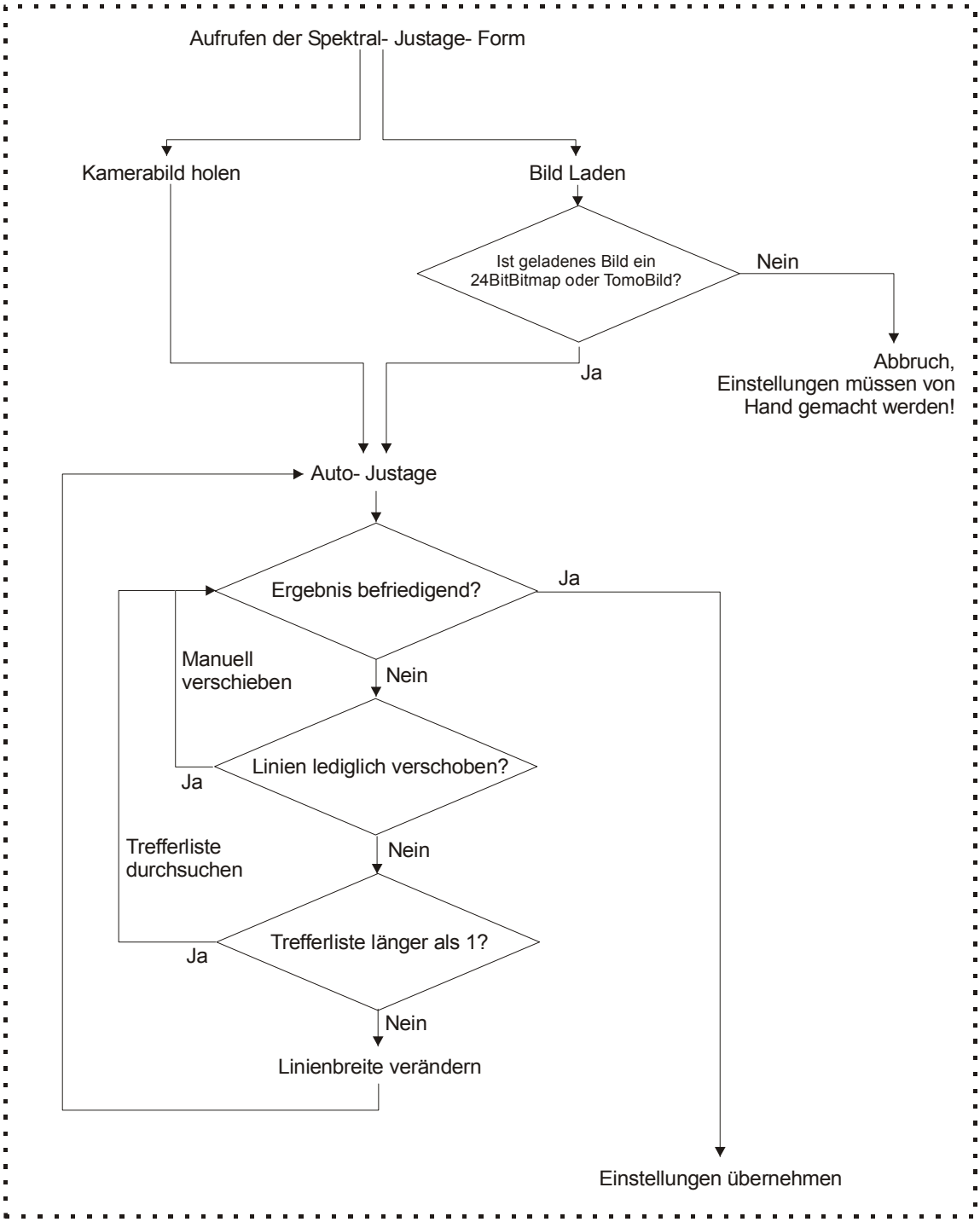


Abbildung 5-25: Einstellmöglichkeiten der Spektral-Justage

Bearbeiten der INI-Datei:
Die Musterspektren müssen damit sie benutzt werden können, in einer INI-Datei abgelegt werden. Diese Datei weist folgenden Aufbau auf:

```
[Verwaltung:]
Anzahl der Elemente=3
Spacer=-------------------------

[0. Element:]
Name=Neon (6032)
Anzahl Linien=3
_0=585.25
_1=594.48
_2=607.43

Spacer=-------------------------

[1. Element:]
Name=Argon
Anzahl Linien=6
_0=453.8
_1=470.4
_2=470.9
_3=471.0
_4=471.2
_5=471.5
```

```
[2. Element:]
Name=Argon (6030)
Anzahl Linien=13
_0=696.54
_1=738.40
_2=750.39
_3=751.47
_4=763.51
_5=772.38
_6=772.42
_7=794.82
_8=801.48
_9=811.53
_10=826.45
_11=840.82
_12=842.46
```

Im Kopf [Verwaltung:] befindet sich ein Eintrag für die Anzahl der in der Datei vorhandenen Spektren x; in diesem Fall ist x=3.
Die nächsten Einträge markieren die einzelnen Spektren. Sie haben immer den selben Aufbau: [0. Element:] bis [(x-1). Element:]!
Im Feld „Name=“ wird dem Spektrum ein Name zugewiesen. Darauf folgt wieder eine Variable für die Anzahl an Spektrallinien, die für dieses Element charakteristisch sind. Die Spektrallinien stehen dann nach Wellenlängen geordnet in der Form _i=000.00 darunter; die Einheit ist [nm]!

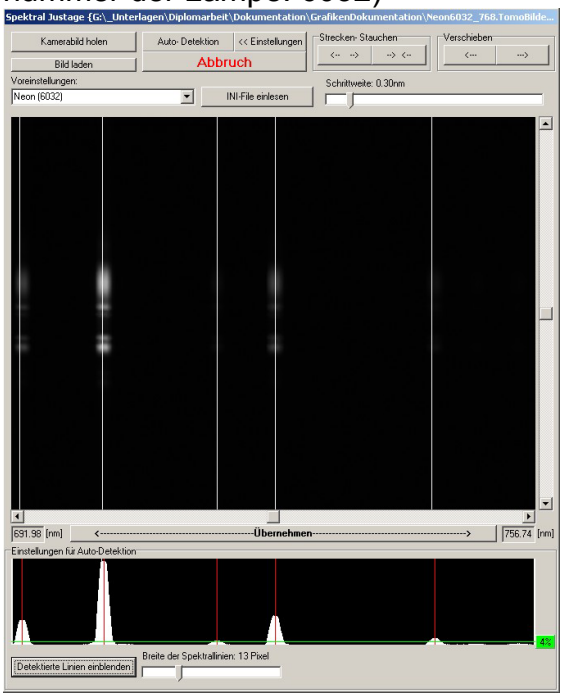
Die INI- Datei kann jeden beliebigen Namen haben, jedoch wird nur die Datei „Spektral_Daten.ini“ bei Programmstart geladen.

Verifizierung der Apparatur:
Die Möglichkeit der Zuordnung von Wellenlängen auf die einzelnen Pixel des Bildes ist für spektroskopische Messungen von großer Wichtigkeit, weshalb die Detektion möglichst genau sein muss. Als Test für die Zuverlässigkeit und der spektralen Genauigkeit der Apparatur wurde dieses Experiment durchgeführt:
Als Referenz dienten zwei Spektrallampen mit unterschiedlichen Wellenlängenbereichen; in diesem Fall waren dies: Neon (Lampennummer 6032) und Argon (Lampennummer 6030). Die Spektren dieser Lampen überschneiden sich im Bereich 692.95[nm] bis 751.47[nm].
Nun wurden nacheinander die beiden Spektren aufgenommen und ausgewertet, wobei bei beiden der gleiche Wert für linke und rechte Wellenlänge herauskommen musste, schließlich wurde der Spektrograph während der Messung nicht verändert!

Das Ergebnis des Test sah folgendermaßen aus:
Die **grobe** Vorgabe, abgelesen aus dem Diagramm (Abbildung 3-15, Seite 21) in „Abbildungseigenschaften der Optik“, war

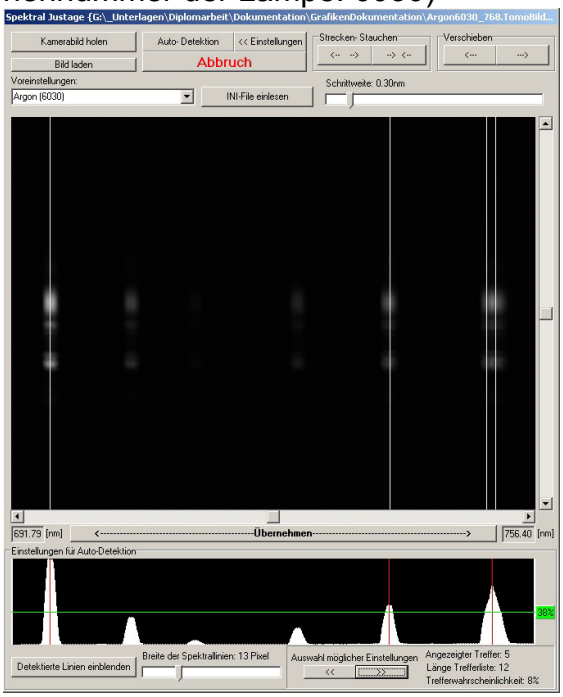
linke Wellenlänge: 690[nm]
Rechte Wellenlänge: 755[nm]
Bei Einstellung Mikrometerschraube 768

Ergebnis Autojustage bei Neon (Seriennummer der Lampe: 6032)



linke Wellenlänge: 691,98[nm]
rechte Wellenl.: 756,74[nm]
Nummer in Trefferliste: 1

Ergebnis Autojustage bei Argon (Seriennummer der Lampe: 6030)



linke Wellenlänge: 691,79[nm]
rechte Wellenl.: 756,40[nm]
Nummer in Trefferliste: 5

Abbildung 5-26: spektrale Verifizierung der Apparatur

Da das Musterspektrum Argon im obigen Bereich lediglich vier Linien umfasste, wovon zwei im aufgenommenen Bild zusammenfielen, erfasste der Algorithmus nicht auf Anhieb die richtigen Wellenlängen. Mit der groben Vorgabe aus dem Diagramm von Seite 21 war jedoch die richtige Lösung schnell an Position fünf in der Trefferliste gefunden.

Fazit:
Der Algorithmus arbeitet richtig. Es kann jedoch vorkommen, dass bei ungünstigen Startbedingungen nicht auf Anhieb die richtige Einstellung gefunden wird. Deshalb sollte immer zur Kontrolle das Diagramm auf Seite 21 herangezogen werden!

Realisierung im Quelltext
Das grundlegende Problem ist das Vergleichen des aufgenommenen Spektralbildes mit dem idealen Spektrum aus der *.INI- Datei. Die Spektrallinien sind sowohl verschoben als auch gestreckt bzw. gestaucht. Zudem liegen sie in unterschiedlichen Einheiten vor: [Pixel] und [nm].
Die einzige Gemeinsamkeit ist das Verhältnis der Abstände von jeweils zwei Linienpaaren.
Die `unit Spektral_AutoDetekt` umfasst alle wichtigen Prozeduren und Funktionen um diese Aufgabe zu bewältigen. `unit Spektral_Justage` beinhaltet die Verwaltung und Reaktion auf Ereignisse.

Zunächst möchte ich das Augenmerk auf `unit Spektral_AutoDetekt` lenken.
Hier sind zwei wichtige Klassen deklariert:

Auszug aus `unit Spektral_AutoDetekt`

```
TMerkmal = class
                        Wert: Single;
                        ersterAbstand: Single;
                        ersteWellenlaenge: Single;
                        //Position: TVektor;
                        end;

TErgebnis = class
                        links: Single;
                        rechts: Single;
                        Vorkommen: Integer;
                        end;
```

Weiterhin werden folgende wichtige Objekte benötigt:

Auszug aus `unit Spektral_AutoDetekt`

```
Bild_Linien:   TList;    //Speichert die Positionen der Spektrallinien des aufgenommenen Bildes

Merkmalsraum:  Tlist;    //Speichert die Merkmale des Musterspektrums

Ergebnisliste: TList;    //Beinhaltet die einzelnen als möglich erachteten Ergebnisse
```

Beim Starten der Autojustage werden aus den beiden zu vergleichenden Spektren bestimmte Merkmale extrahiert, welche im Weiteren miteinander verglichen werden.
Kommt es zu einer Übereinstimmung, so werden die dazu passenden Wellenlängen der linken und rechten Bildkante errechnet und in einer Liste (`Ergebnisliste`) als TErgebnis gespeichert.
TErgebnis beinhaltet die Anzahl, in der es mit seinen charakteristischen Wellenlängen detektiert wurde im Integer Vorkommen. Somit werden in die Ergebnisliste aus-

schließlich Ergebnisse mit unterschiedlicher linker und rechter Wellenlänge eingetragen, jedes einzelne Ergebnis speichert sein Vorkommen.
Über Vorkommen kann später das wahrscheinlichste Ergebnis ermittelt werden: Je höher dieser Wert ist, um so wahrscheinlicher ist eine richtige Detektion.
Sind zum Beispiel Musterspektrum und aufgenommenes Spektrum identisch, so sind auch deren Merkmale identisch und die Ergebnisliste besteht aus nur einem Element.

Um den gesamten Algorithmus besser verstehen zu können, sei folgender Fall angenommen:
Die Merkmale eines aufgenommenen Bildes sollen bestimmt werden und mit den Merkmalen des idealen Spektrums, bestehend aus vier Linien, verglichen werden:

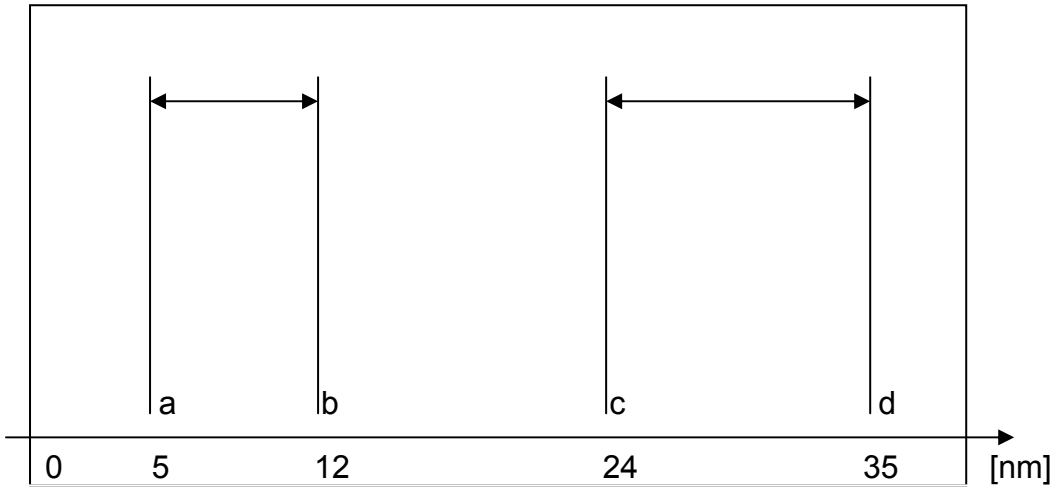


Abbildung 5-27: Beispiel für ein aufgenommenes Bild

Die Linien sind durch ihre Wellenlängen eindeutig bestimmt. Prozedur `Merkmalsraum_fuellen` errechnet daraus alle Merkmale und speichert sie im Merkmalsraum.
Es werden jeweils zwei mögliche ungleiche Paare aus aufeinander folgenden Linien betrachtet und deren Abstände zueinander ins Verhältnis gesetzt:

Auszug aus `Merkmalsraum_fuellen`

```
Merkmal.Wert:=(b.w-a.w)/(d.w-c.w);
Merkmal.ersterAbstand:=b.w-a.w;
Merkmal.ersteWellenlaenge:=a.w;
Merkmalsraum.Add(Merkmal);
```

Merkmal.Wert:= $\frac{b.w - a.w}{d.w - c.w} = \frac{12-5}{35-24} = 0,\overline{636}$

Dies ist der charakteristische Wert des Merkmals, der mit den Werten der anderen Merkmale verglichen wird.

Merkmal.ersterAbstand:= *7[Pixel]*

Merkmal.ersteWellenlänge:= *5 [Pixel]*

Diese Einträge werden erst bei Übereinstimmung mit einem Merkmal des aufgenommenen Spektrums benötigt. Mehr dazu siehe unten.

Nachdem der Merkmalsraum für das Muster-Spektrum gefüllt ist, beginnt die Prozedur `vergleichenUndWerten()`. Sie führt im Grunde genommen das gleiche mit dem aufgenommenen Spektrum durch, wie `Merkmalsraum_fuellen` mit dem Muster-Spektrum, nur vergleicht sie sofort das berechnete Merkmal mit allen Merkmalen im Merkmalsraum ohne weitere Zwischenspeicherung.
Hier kommt die vom Benutzer eingestellte Linienbreite ins Spiel:
Da die aufgenommenen Spektrallinien nicht ideal dirac-förmig sind, sondern eher einem Gaußprofil ähneln, wurde deren Position durch das Flächengewicht dieses Profils errechnet. Dies birgt natürlich gewisse Fehler, weshalb beim Vergleich mit den idealen Daten eine Toleranz (`Offset`) eingeführt werden muss:

Auszug aus vergleichenUndWerten

```
if (b.w+Offset<>a.w-Offset) and (d.w-Offset<>c.w+Offset) and (b.w-
Offset<>a.w+Offset) and
      (d.w+Offset<>c.w-Offset) and (b.w<>a.w) and (d.w<>c.w) then
begin
              Maxwert:=((b.w+Offset)-(a.w-Offset))/((d.w-Offset)-(c.w+Offset));
              Minwert:=((b.w-Offset)-(a.w+Offset))/((d.w+Offset)-(c.w-Offset));
              if not Abbruch then
                     for x:=0 to Merkmalsraum.Count-1 do
                     begin
                            tmp:=Merkmalsraum[x];
                            if (tmp.Wert<Maxwert) and (tmp.Wert>Minwert) then
                            begin
                                   writeToErgebnisliste(a.w,b.w-a.w,tmp);
                            end;
                     end;
end;
```

`tmp` ist ein TMerkmal, das jeweils aus dem Merkmalsraum extrahiert und mit den gerade berechneten Werten `Maxwert` und `Minwert` verglichen wird.
Sollte sich eine Übereinstimmung ergeben, wird der Ergebnisliste ein neues Ergebnis hinzugefügt, was `writeToErgebnisliste(a.w,b.w-a.w,tmp)` leistet.

In `writeToErgebnisliste(ersteLinie, ersterAbstand: Single; Merkmal: TMerkmal)` wird zunächst ein neues Ergebnis kreiert und dessen linke und rechte Wellenlänge errechnet:

```
Ergebnis.links:=Merkmal.ersteWellenlaenge-ersteLinie*Merkmal.ersterAbstand/ersterAbstand;
Ergebnis.rechts:=Ergebnis.links+639*Merkmal.ersterAbstand/ersterAbstand;
```

Eine Einheitenkontrolle zeigt, dass korrekt von Pixelposition in Wellenlänge umgerechnet wird: $\text{Ergebnis.links} = [nm] - \frac{[Pixel]\cdot[nm]}{[Pixel]} = [nm].$

Nun wird `Ergebnis.Vorkommen` auf 1 gesetzt und verglichen, ob es bereits einen Eintrag in der Ergebnisliste mit den Wellenlängen der linken und rechten Seiten gibt.
Allerdings ist es unwahrscheinlich, einen Eintrag in der Ergebnisliste zu finden, der exakt passt, da dieser ja bereits aus mit Messfehlern und Rundungsfehlern belasteten Zahlen errechnet wurde. Deshalb ist ein weiterer Toleranzfaktor `d=1.0[nm]` nötig. Das Problem hierbei ist, dass `d` im Endeffekt die Genauigkeit der Messapparatur bestimmt. Wird er zu klein gewählt, so resultieren daraus zu viele Einträge in der Ergebnisliste; die Wahrscheinlichkeit das richtige Ergebnis anzuzeigen, würde sich verringern. Eine Vergrößerung verschlechtert allerdings die Genauigkeit. Deshalb wur-

den einige Testmessungen an definierten Spektren durchgeführt, wobei sich der Faktor `d=1.0[nm]` als optimal ergab.

Noch einmal zusammenfassend: Das aufgenommene Spektrum wurde mit dem in der *.INI- Datei abgelegten Musterspektrum verglichen. Ergaben sich Übereinstimmungen, so wurde errechnet, welche linke und rechte Wellenlänge im Bild sein müsste. Diese Werte wurden dann in einer Ergebnisliste abgelegt, wobei schon vorhandene Einträge in ihrem Vorkommen erhöht wurden. Die Ergebnisliste besteht also aus ungeordneten möglichen Einstellungen für linke und rechte Wellenlänge; einige von ihnen wurden vom Algorithmus öfters die anderen weniger oft als Lösung erachtet. Ordnet man nun alle Einträge nach ihrem Vorkommen, so erhält man eine Liste, bei deren Durchlaufen man von der wahrscheinlichen zur eher unwahrscheinlichen Lösung kommt.
Dies erledigt die Prozedur `ErgebnislisteAuswerten()`, die aus einem einfachen Sortieralgorithmus besteht.
Danach wird dann die wahrscheinlichste Lösung (Position 0 in der Ergebnisliste) auf dem Bildschirm mit `ErgebnisAnzeigen(Stelle: Integer)` ausgegeben.

Die folgende Grafik soll den gesamten Algorithmus noch einmal grafisch veranschalichen:

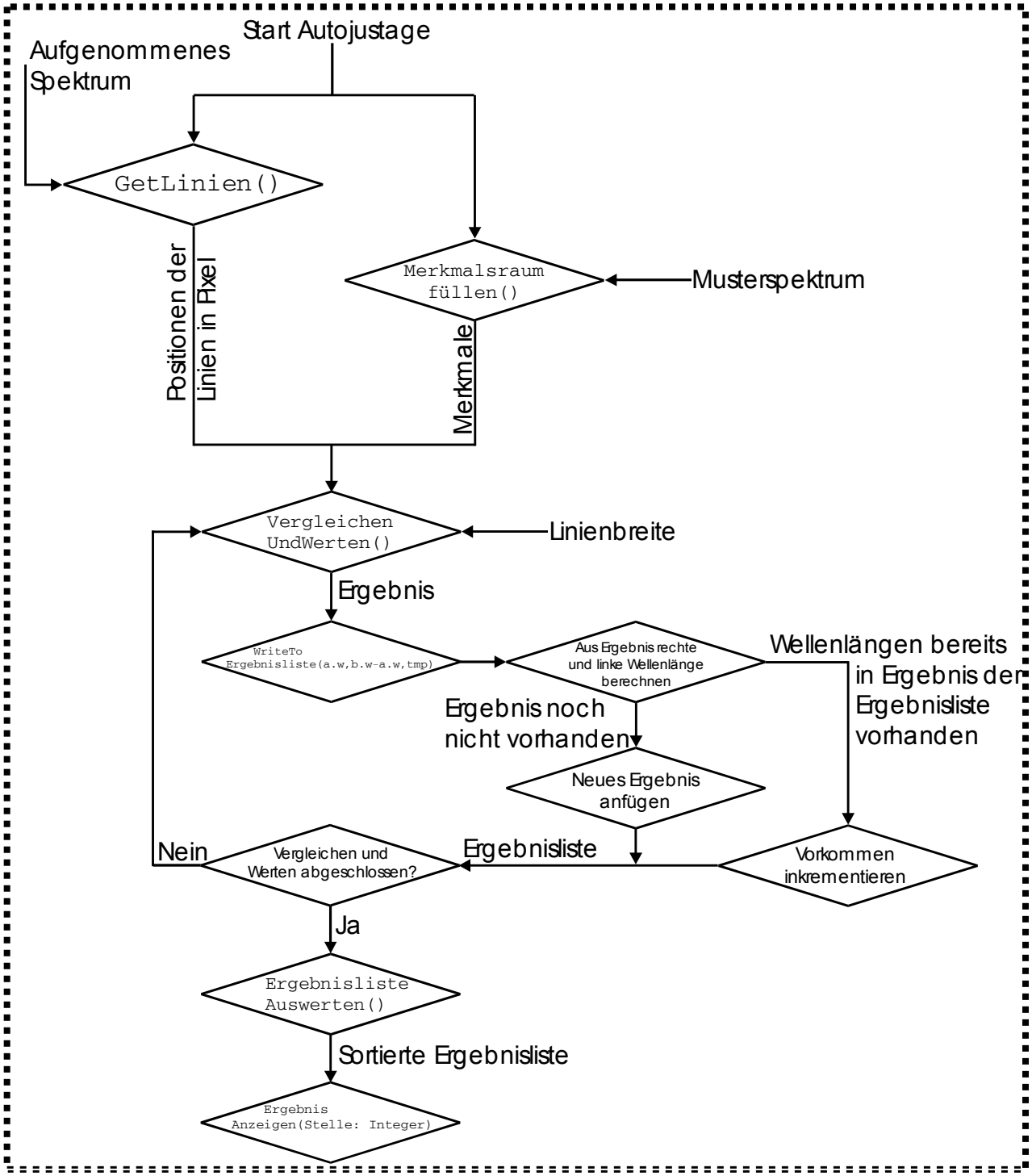


Abbildung 5-28: Ablaufdiagramm des Algorithmus zur Spektraljustage

Abschätzung des Rechenaufwandes bei einem Musterspektrum mit n Linien und einem aufgenommenen Spektrum mit m Linien:

Sei n die Anzahl der Linien des Musterspektrums, so umfasst der Merkmalsraum $(n-1)(n-2)$ Merkmale.

Im obigen Beispiel mit vier Linien sind somit folgende sechs Verhältnisse enthalten:

$$\frac{b-a}{c-b}, \frac{c-b}{b-a}, \frac{d-c}{b-a},$$
$$\frac{b-a}{d-c}, \frac{c-b}{d-c}, \frac{d-c}{c-b}$$

Im aufgenommenen Spektralbild werden genauso $(m-1)(m-2)$ Merkmale extrahiert.
Aus diesen Merkmalen wird jeweils ein Maximal- und Minimalwert errechnet, der wiederum mit jedem Merkmal des Musterspektrums verglichen wird.
Dies ergibt also $(n-1)(n-2)\cdot(m-1)(m-2)\cdot 2$ Vergleichsoperationen.
(Sei $n=4$ wie oben und $m=5$ dann sind dies bereits 144 Vergleichsoperationen)

5.4.8 Das Formular „Grafische Überwachung“

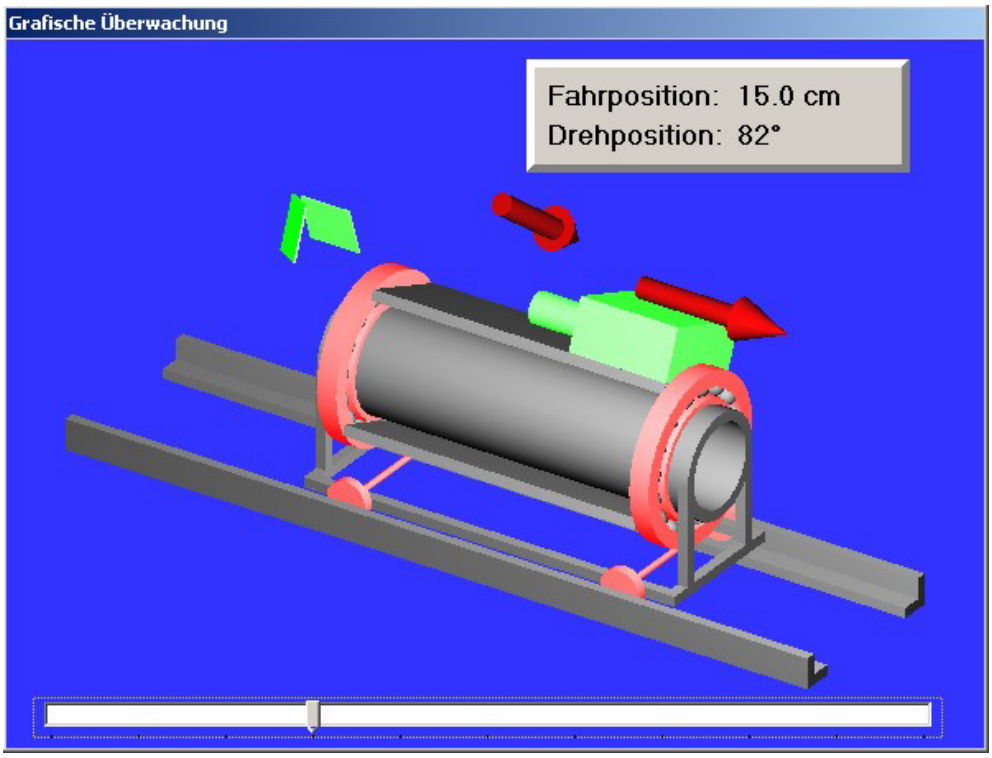


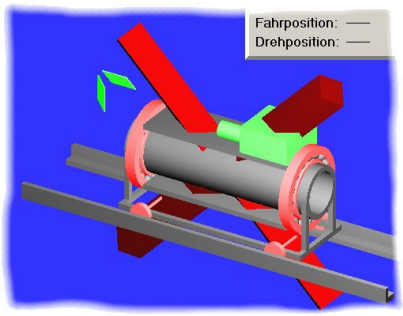
Abbildung 5-29: Das Formular "Grafische Überwachung"

Beschreibung:
Das Formular Überwachung dient, wie der Name schon sagt, der Überwachung der gesamten Messanordnung. Alle wichtigen Zustände, sowie die im Rechner gespeicherten Positionen der Apparatur können hier auf einen Blick eingesehen werden.
Gelegentlich kann es vorkommen, dass die tatsächlichen von den im Rechner gespeicherten Positionswerten abweicht –zum Beispiel nach unachtsamem Auslösen des Anschlags-Tasters.
Diese im allgemeinen gravierenden Abweichungen können mit Hilfe der Überwachung sofort erkannt werden.

Zustände:
Bis jetzt sind fünf verschiedene Zustände realisiert, weitere können jedoch ohne großen Aufwand an neue Gegebenheiten angepasst werden.

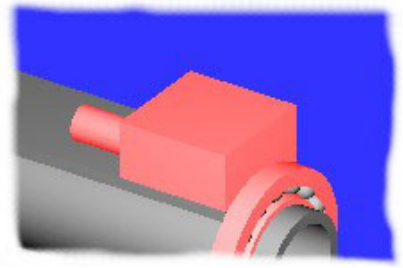
Rote Pfeile:
Sie zeigen die Richtung an, in die sich die Apparatur bewegt.

Motor nicht referenziert:



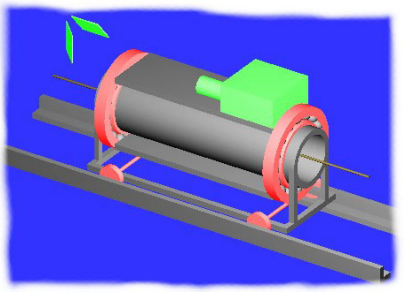
Die Steuerung wurde noch nicht referenziert:
Mit *Referenzfahrt* oder *Positionsabfrage* Daten mit Interface abgleichen!

Kameraprobleme:



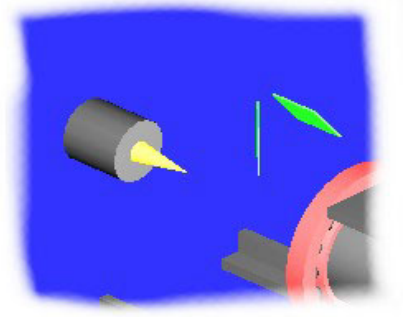
Die Kamera wurde noch nicht initialisiert, beziehungsweise sie ist überhitzt!

Drehachsjustage:



Im Bild ist ein Draht in der Messtrommel dargestellt. Entweder die Drehachse ist noch nicht eingestellt oder die Drehachsjustage läuft gerade ab.

Messung:



Die Messung läuft gerade ab, was durch den Plasmabrenner dargestellt wird. Eine frühere Version der Akina-Software beinhaltet für die Plasmaflamme anstatt des gelben Kegels einen GLFireFXManager des OpenGL-Packets. Dieser ist zwar optisch sehr ansprechend, benötigt aber viel kostbare Rechenzeit. Aus diesem Grunde wurde er durch den Kegel ersetzt. Überhaut ist es während einer Messung gerade bei langsamen Rechnern zu empfehlen, das Formular Überwachung zu schließen.

Abbildung 5-30: Übersicht über die Zustände

Realisierung im Quelltext:
Das Herz dieser Form ist wie auch bei dem 3D-Browser das OpenGL-Packet, das alle grafischen Darstellungen übernimmt.
Interessant für die Darstellung der verschiedenen Zustände ist die Klasse `TStatusAufbau` in Globals. Da die Überwachungs- Form erst dann erstellt wird, sobald sie auf dem Bildschirm dargestellt werden soll, müssen die einzelnen Zustände in dieser Klasse extern gespeichert werden:

Ausschnitt aus `Globals`:

```
type    TQuelle = (INIT,vonGlobals,Motor,Kamera,Drehachse,Messung);

        TStatus = record
                Quelle: TQuelle;
                MotorReferenziert: Boolean;
                KameraInitialisiert: Boolean;
                Drehachseeingestellt: Boolean;
                MessungLaeuft: Boolean;
                end;

        TBewegungsStatus = record
                PosX, PosY: Integer;
                ZielX, ZielY: Integer;
                GeschwX, GeschwY: Integer;
                end;

        TStatusAufbau = class
        public
                StatusAktuell: TStatus;
                Bewegungsstatus: TBewegungsstatus;
                procedure WriteBewegungsStatus(const Value: TBewegungsstatus);
                procedure MotorbewegungSTOP();
                procedure WriteStatus(const Value: TStatus);
        end;
```

In `TStatus` sind alle Zustände gespeichert. Da jedoch die Motorsteuerung lediglich `MotorReferenziert` verändern darf usw., beinhaltet `TStatus` auch die Quelle, von der die Anforderung zur Änderung des Zustands kommt.
`TBewegungsStatus` wird nur von der Motorsteuerung verändert, hier wird der Bewegungszustand festgehalten.

5.5 Datenverarbeitung

5.5.1 Das Formular „Tomografische Auswertung“

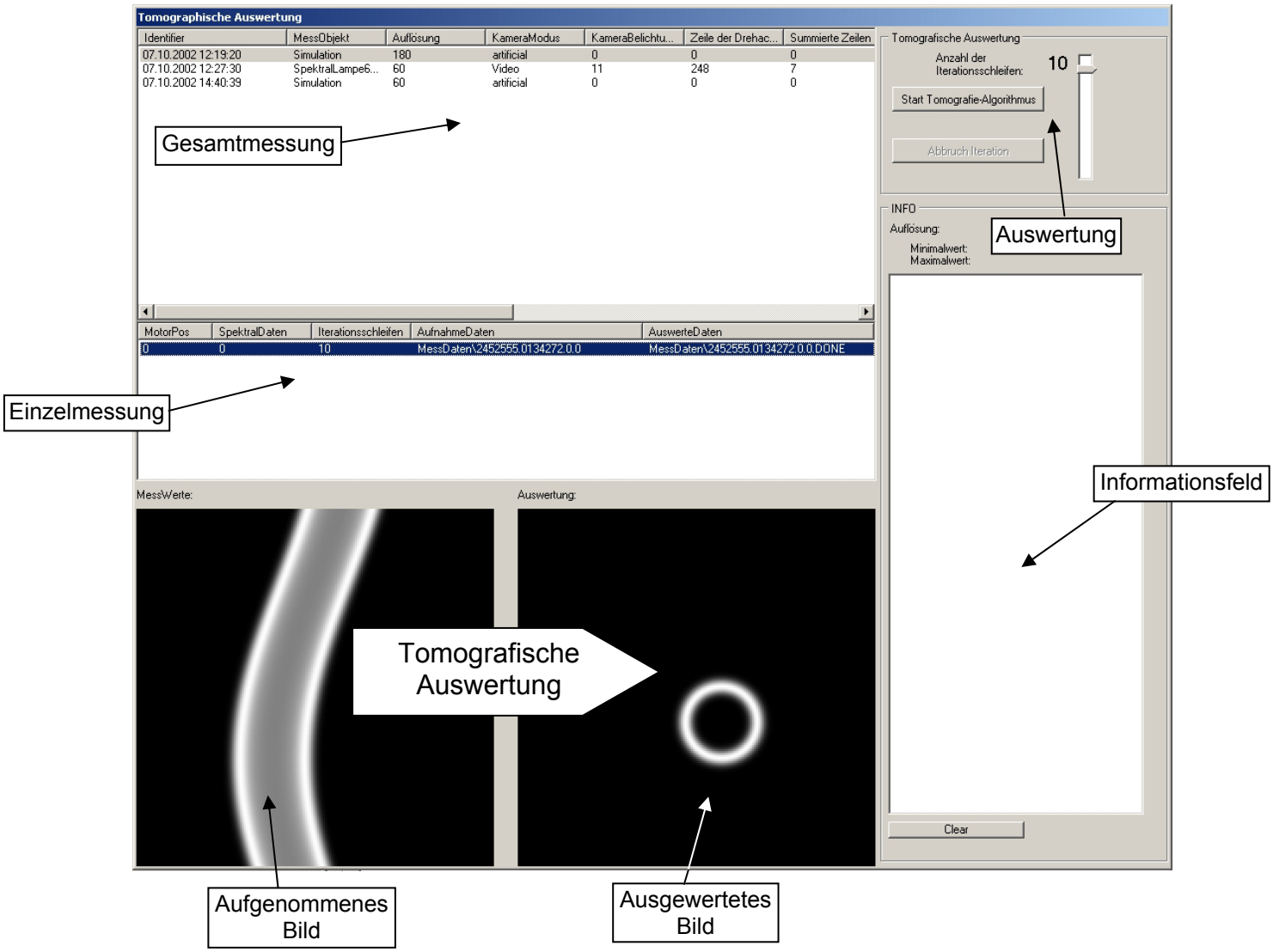


Abbildung 5-31: Formular tomographische Auswertung

Das Formular tomographische Auswertung zeigt alle aufgenommen Messungen, sowie deren Auswertungen (falls vorhanden) an. Im oberen Teil werden hier die Gesamtmessungen angezeigt. Hier trifft der Nutzer eine Vorauswahl zu welcher Messung er nun die einzelnen Aufnahmen sortiert, nach Motorposition und Spektrallinie auswerten bzw. nur ansehen will. Ist dann im Feld Einzelmessung mindestens eine der Einzelaufnahmen markiert, werden die enthaltenen Daten darunter in den beiden Bildern angezeigt (soweit schon ein ausgewertetes Bild zur Verfügung steht). Sind mehrere Datensätze markiert, so wird nur das zuerst angewählte dargestellt.
Nachdem im Bereich Auswertung die Anzahl der Iterationsschleifen eingestellt wurden, kann mit der Auswertung durch Klick auf „Start Tomografie – Algorithmus“ begonnen werden. Die Datensätze werden jetzt der Reihe nach ausgewertet. Im Informationsfeld lässt sich während der Auswertung deren Fortschritt erkennen. Erst wenn ein Datensatz komplett ausgewertet ist wird er in der Datenbank gespeichert. Besonders bei hohen Auflösungen (>120) benötigt der Rechner zunehmend Arbeitsspeicher und Rechenzeit. Wird die Auswertung abgebrochen durch Klick auf die

Schaltfläche „Abbruch Iteration“, wird die laufende Iteration noch durchgeführt, nach ihr der Algorithmus jedoch beendet. Die Daten werden dann nicht in der Datenbank gespeichert und gehen verloren.
Obwohl auf Grund der Grafikkartenarchitektur nur eine Darstellung der Messung in einer acht Bit Tiefe möglich ist, werden dennoch zwölf Bit Werte für die Auswertung verwendet und auch wieder zur späteren Analyse gespeichert.
Im Informationsfeld befindet sich noch zusätzlich eine Anzeige Minimal- bzw. Maximalwert. Sie gibt dem Benutzer während der Auswertung Informationen über den Maximal- bzw. Minimalwert der Auswertung.
Um nicht mehr benötigte Datensätze wieder entfernen zu können ist es möglich, diese durch Markieren und anschließendes Drücken der Taste „Entf“ zu löschen. Die Daten gehen dabei unwiderruflich verloren.
Die genaue Funktionsweise des Tomografiealgorithmussees wurde bereits in einer am Institut EIT 2 angefertigten Diplomarbeit ausreichend beschrieben und soll deshalb hier nur sehr kurz erläutert werden [2, 6].

5.5.2 Das Formular „Serienbildbrowser“

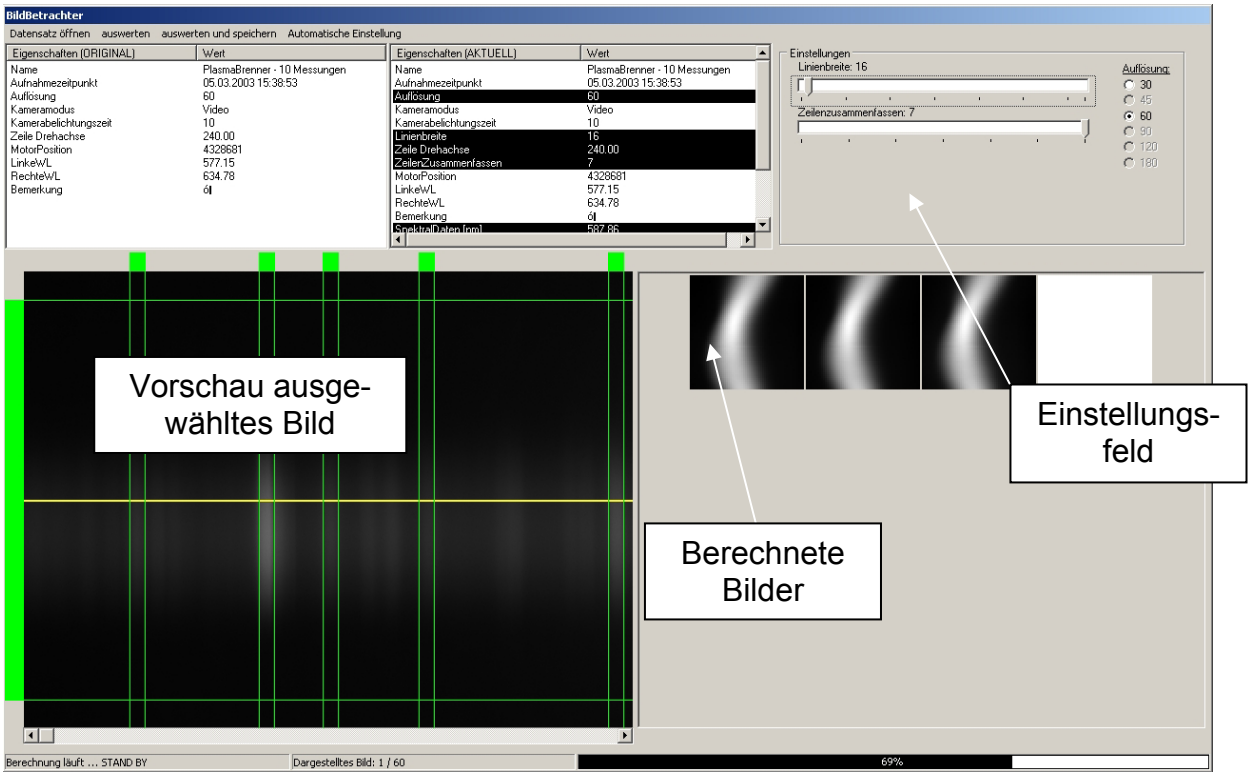


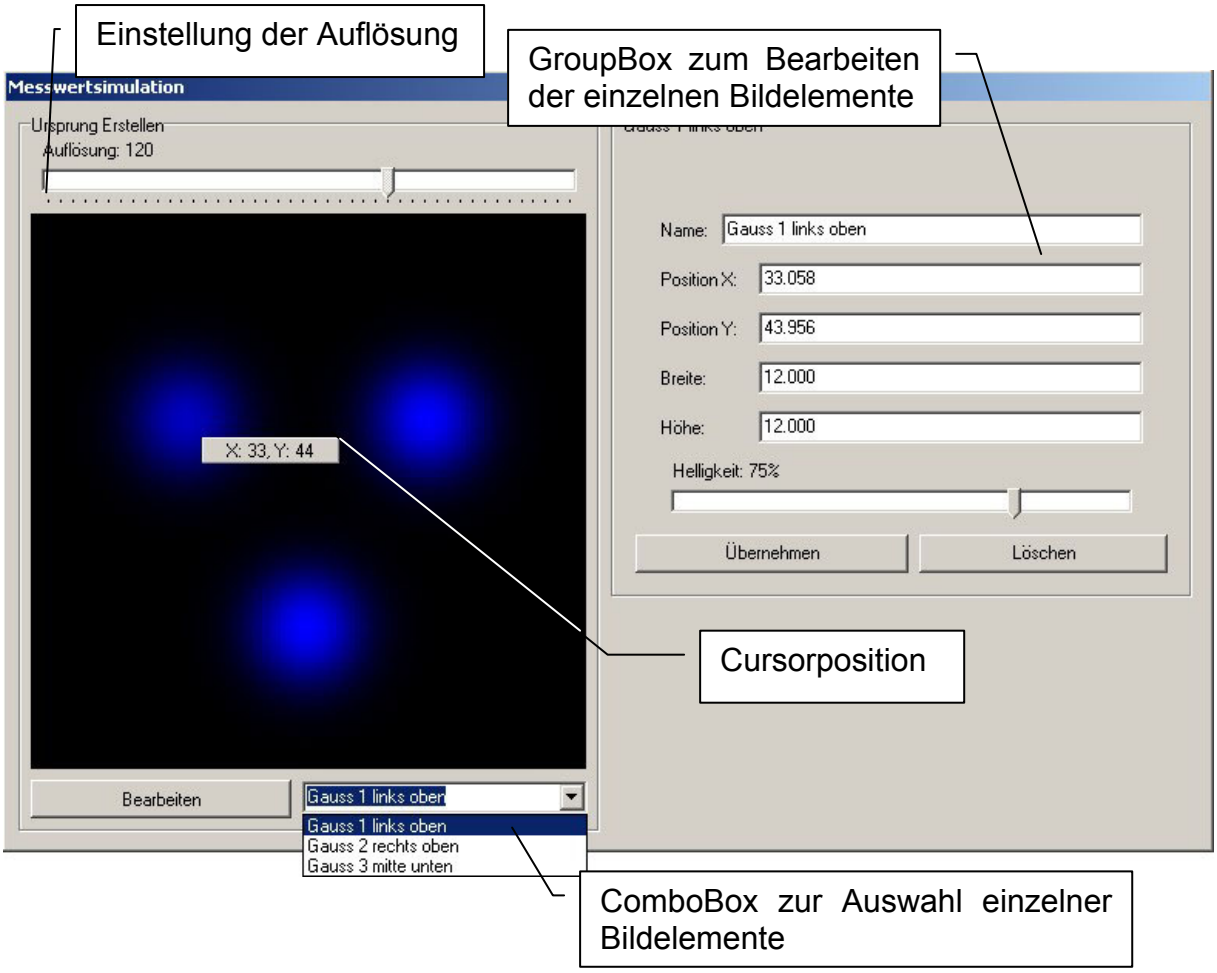
Abbildung 5-32: Das Formular Serienbildbrowser

Es besteht die Möglichkeit alle Daten schon während der Messung zu verarbeiten. Um die aufgenommenen 12Bit Bilder nach der Messung noch zusätzlich auswerten zu können, wurde dieses Formular entwickelt. Es kann auf Wunsch alle aufgenommenen Bilder einzeln anzeigen. Ähnlich wie in bereits bestehenden Formularen, besteht die Möglichkeit einzelne Spektrallinien auszuwählen und zuspeichern. Alle notwendigen Parameter sind rechts oben eingestellbar. Die Informationen über Aufnahmezeitpunkt und Einstellungen der Messapparatur werden aus dem Header der eingelesenen Datei in die Datenbank übernommen. Eine Vorschau des ausgewerteten Bildes steht ebenfalls zur Verfügung.
Wurden die Bilder mit einer sehr hohen Auflösung aufgenommen, können niedrigere Auflösungen ebenfalls ausgewertet werden.

Die `Scrollbar` unter dem Vorschaubild zeigt alle Bilder der Reihe nach auf dem Formular links oben in Originalgröße an. Auf diese Weise können Aufnahmefehler (Instabilitäten der Quelle, etc.) festgestellt werden.

Damit dieses Formular eingesetzt werden kann, müssen alle Bilder während der Messung aufgenommen worden sein. Da dabei große Datenmengen entstehen, geschieht dies nicht automatisch.
Eine Auswertung in einer anderen als der aufgenommenen Auflösung ist nur möglich, wenn die neue Auflösung ganzzahlig durch die alte teilbar ist. Nur dann entstehen keine Auswertungsfehler, da für den Tomografiealgorithmus Aufnahmen in einem Sektor von 180° gleichverteilt vorliegen müssen. Überzählige Aufnahmen werden dabei einfach übersprungen.
Die Menuleiste im Kopfbereich des Formulars gibt die Möglichkeit einen neuen Datensatz zu laden, die Bilder auszuwerten und zu speichern.

5.5.3 Das Formular „Simulation tomografischer Messdaten“



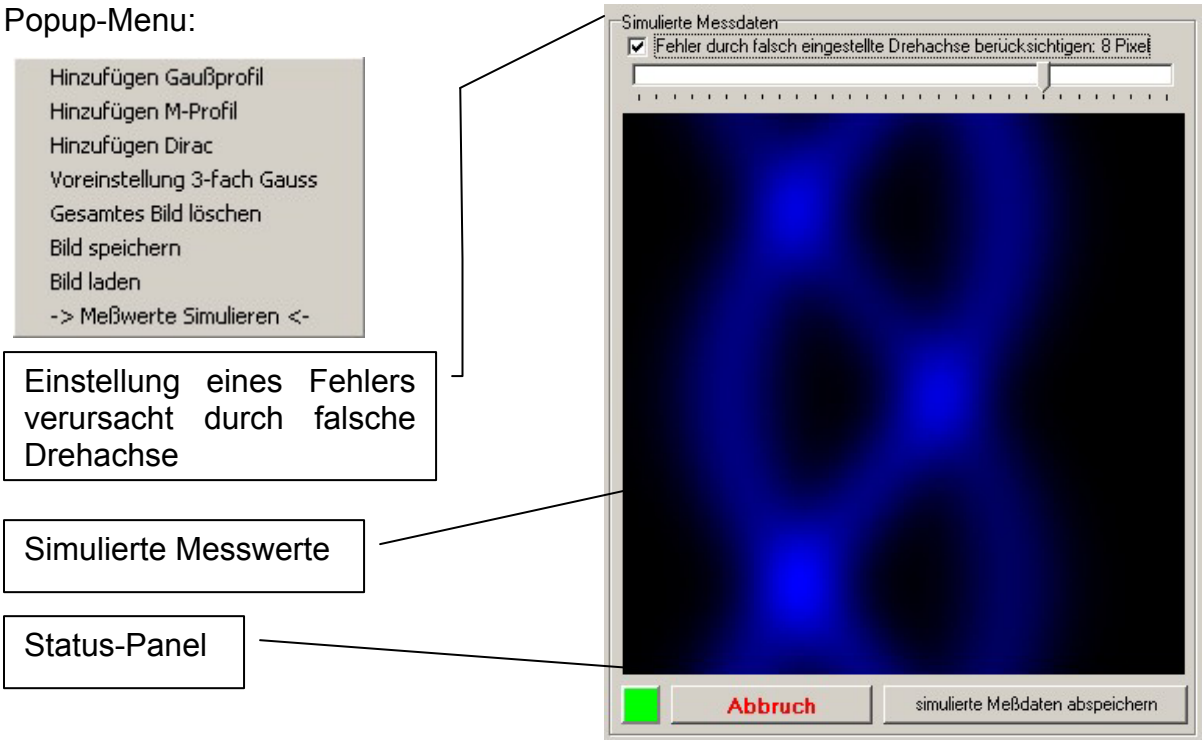


Abbildung 5-33: Das Formular "Simulation tomografischer Messdaten"

Beschreibung:
Ein realer Messablauf ist normalerweise sehr aufwendig: Zuerst müssen Hardware eingestellt, Drehachsen justiert und Messungen durchgeführt werden, bis erste Werte zur Verfügung stehen. Möchte man jedoch nur die Arbeitsweise des Algorithmus testen, so bietet sich an, idealtypische Messwerte vom Computer erzeugen zu lassen. Idealtypisch deshalb, da jegliche Messfehler sowie Ungenauigkeiten der Optik keine Rolle spielen. Vielmehr können einige in der Realität auftretende Messfehler definiert eingestellt werden. All diese Möglichkeiten bietet die Messwertesimulation im Akina-Projekt.

Erstellen von Simulierten Messwerten:
Grundsätzlich gibt es drei verschiedene „Bildelemente“:

- Dirac
- Gauss
- M-Profil

Sie stellen die Emission einer Wellenlänge bei einem bestimmten Schnitt durch ein virtuelles, strahlendes Objekt dar und können beliebig miteinander kombiniert werden.
Zusätzlich steht eine Voreinstellung, der dreifach-Gauss, zur Verfügung. Er soll die Emission eines Triplex-Brenners simulieren.

Um Messwerte zu simulieren, muss man zunächst die gewünschte Auflösung an der Track-Bar einstellen; sie bildet die Grundlage.
Nach Rechtsklick auf das Image oder Betätigung des Bearbeiten-Buttons öffnet sich das Popup-Menu, an dem alle Bildelemente sowie Voreinstellungen geladen werden können. Durch Linksklick auf das Image werden schließlich die Elemente platziert.
Über das Popup-Menu kann zusätzlich ein fertiggestelltes Bild gespeichert beziehungsweise ein altes Bild geladen werden, was sich gerade bei aufwendig gestalteten Bilder mit vielen Elementen anbietet.
Nachdem das gewünschte Bild erstellt wurde, können die Messwerte simuliert werden - hierzu im Popup-Menu →Messwerte simulieren← betätigen.
Ein Klick auf Flächenfaktoren berechnen → Messwerte simulieren startet den Algorithmus. Vorsicht: Je höher die Auflösung, um so rechenaufwendiger ist der Prozess; es kann also je nach Ausstattung des Computers einige Zeit in Anspruch nehmen.
Nun kann noch je nach Wunsch ein Fehler durch eine falsch eingestellte Drehachse erzeugt werden, bevor dann die Daten unter der Bezeichnung „Simulation“ in die Datenbank übernommen werden.

Realisierung im Quelltext

Grundsätzlich ist die Messwertesimulation in zwei Units unterteilt. `unit Simulation` übernimmt im Wesentlichen die Ereignisbehandlung, wobei `unit Simubuilder` die Berechnungen sowie das Erstellen der Images umfasst.
Folgende Typen wurden definiert:

```
TFunktion = (Gauss, MProfil, Dirac);
TContent = Class
             Art: TFunktion;
             PositionX:String;
             PositionY:String;
             DX:String;
             DY:String;
             Helligkeit: Single;
             Name:String;
             end;
```

TFunktion beschreibt die Art eines Bildelements, und TContent ist eine Klasse, in der Art, Position und Aussehen sowie Helligkeit (relativ) und Bezeichner gespeichert werden. Position, DX und DY sind deshalb Strings, da sie in den Edit-Feldern und in der INI-Datei als Strings behandelt werden. Die Umwandlung zum numerischen Wert erfolgt erst beim Erstellen der Grafik.

Im Weiteren werde ich die Begriffe „Originalbild“ und „Messwertebild“ benutzen: Originalbild ist das Image, in dem das virtuelle zu untersuchende Objekt erstellt wird; Messwertebild zeigt dann die daraus simulierten Messwerte.

Nun zum Vorgehen beim Erstellen eines Bildes:
Äußert der Benutzer den Wunsch nach einem neuen Bildobjekt durch Betätigen einer Schaltfläche im Popup-Menu, wird zunächst lediglich von den Prozeduren `newGaussClick`, `newMClick` und `newDiracClick` die `EditBox` (eine GroupBox) in der rechten Hälfte der Form dargestellt, jeweils mit den benötigten Edit-Feldern.
Die Erzeugung eines neuen Eintrages TContent in der Liste `eListe` erfolgt erst nach Betätigung des Übernehmen- Buttons.
In `eListe` sind also alle Bildelemente mit ihren Rahmendaten abgelegt.
Das Erstellen des Originalbildes übernimmt nun, wie oben genannt, die `unit Simubuilder` mit der Prozedur `Bild_berechnen_malen()`:

Ausschnitt aus `Bild_berechnen_malen()`

```
for i:=0 to Aufloesung-1 do
   for j:=0 to Aufloesung-1 do
   begin
       Org := TEintrag.Create; Org.w:=0;
       Bild_Original.Add(Org);
   end;
```

Dieser Auszug löscht zunächst das aktuelle Bild und setzt den Eintrag jedes Pixels auf null (=schwarz). Danach werden die einzelnen Bildelemente nacheinander eingefügt. Das Vorgehen bei einem Gaussprofil sei hier stellvertretend für alle weiteren Bildelemente erklärt.

Ausschnitt aus `Gauss_Berechnen()`

```
for i:=0 to Aufloesung-1 do
   for j:=0 to Aufloesung-1 do
   begin
      Org := Bild_Original.Items[Aufloesung*i+j];
      if ((DX>0) AND (DY>0) then
         Org.w := Org.w + e.Helligkeit * exp(
                     -sqr( (j-X) / (DX) )
                     -sqr( (i-Y) / (DY) )   )
         else Org.w := Org.w + 0;
   end;
```

Die Variablen DX und DY geben die horizontale und vertikale Ausdehnung des Gauss an, X und Y die Position im Bild in Pixeln.
Es kann vorkommen, dass sich einzelne Bildelemente überlagern. In der Realität würde dies die Addition der Emissionswerte bedeuten - dies muss also auch eine Simulation leisten: `Org.w` der Helligkeitswert eines Pixels wird deshalb immer um den neu hinzukommenden Wert erhöht.
Die Normierung auf den maximalen im Image anzeigbaren Helligkeitswert 255 erfolgt später in der Prozedur `Bitmap_Erstellen()`.

Sind alle Bildelemente in dieser Art in das Originalbild (Liste `Bild_Original`) übernommen, werden in `Bild_Original_Beschneiden()` die Pixel auf 0 (=schwarz) gesetzt, die der Tomoalgorithmus nicht berücksichtigt (siehe hierzu Abbildung 5-34).
`Bitmap_Erstellen()` erzeugt schließlich das im Originalbild angezeigte Bitmap.

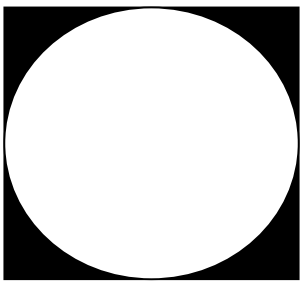


Abbildung 5-34: zulässiger Bereich

Ausschnitt aus `Bild_Original_Beschneiden()`

```
for i:=0 to Aufloesung-1 do
   for j:=0 to Aufloesung-1 do
      if sqrt( sqr(i-(Aufloesung-1)/2) + sqr(j-(Aufloesung-1)/2) )
         >= ((Aufloesung-1)/2) then
         begin
            Org := Bild_Original.Items[Aufloesung*i+j];
            Org.w := 0;
         end;
```

Die Betätigung des Buttons Flächenfaktoren berechnen → Messwerte simulieren startet die Prozedur `Flaechenfaktoren_berechnen(n, w)` in der `unit Tomografie`. Im Anschluss daran werden mit Hilfe der berechneten Flächenfaktoren die Projektionen aus verschiedenen Richtungen des Objektes simuliert und in die Liste `Messwerte_Simuliert` abgelegt
(Prozedur `Messwerte_Simuliert_Berechnen` im `Simubuilder`).
Die genaue Realisierung dieses Vorgangs sei hier nicht weiter erläutert. Sie ist Bestandteil einer eigenen Diplomarbeit und kann dort nachgelesen werden [2, 6].

Abschließend möchte ich noch auf die Erstellung der Items-Datei .itm eingehen:
Im Endeffekt verbirgt sich dahinter eine INI-Datei; es ist die einfachste Möglichkeit, Daten dieser Art zu speichern. Außerdem lässt eine INI-Datei die Bearbeitung über einen einfachen Texteditor zu.

Prozedur `BildspeichernClick` in `unit Simulation` übernimmt diese Aufgabe und erzeugt eine Datei in der Form:

Kopf:	Inhalt:

`[Verwaltung:]`
`Erstellungsdatum=15.01.2003 22:13:43`
`Anzahl der Elemente=3`
`Auflösung=60`
	`[0. Element:]`
	`Name=Gauss 1`
	`Typ=Gauss`
	`Helligkeit=1`
	`Position X=16.529`
	`Position Y=21.978`
	`DX=6.000`
	`DY=6.000`

5.5.4 Das Formular „Spektroskopische Auswertung“

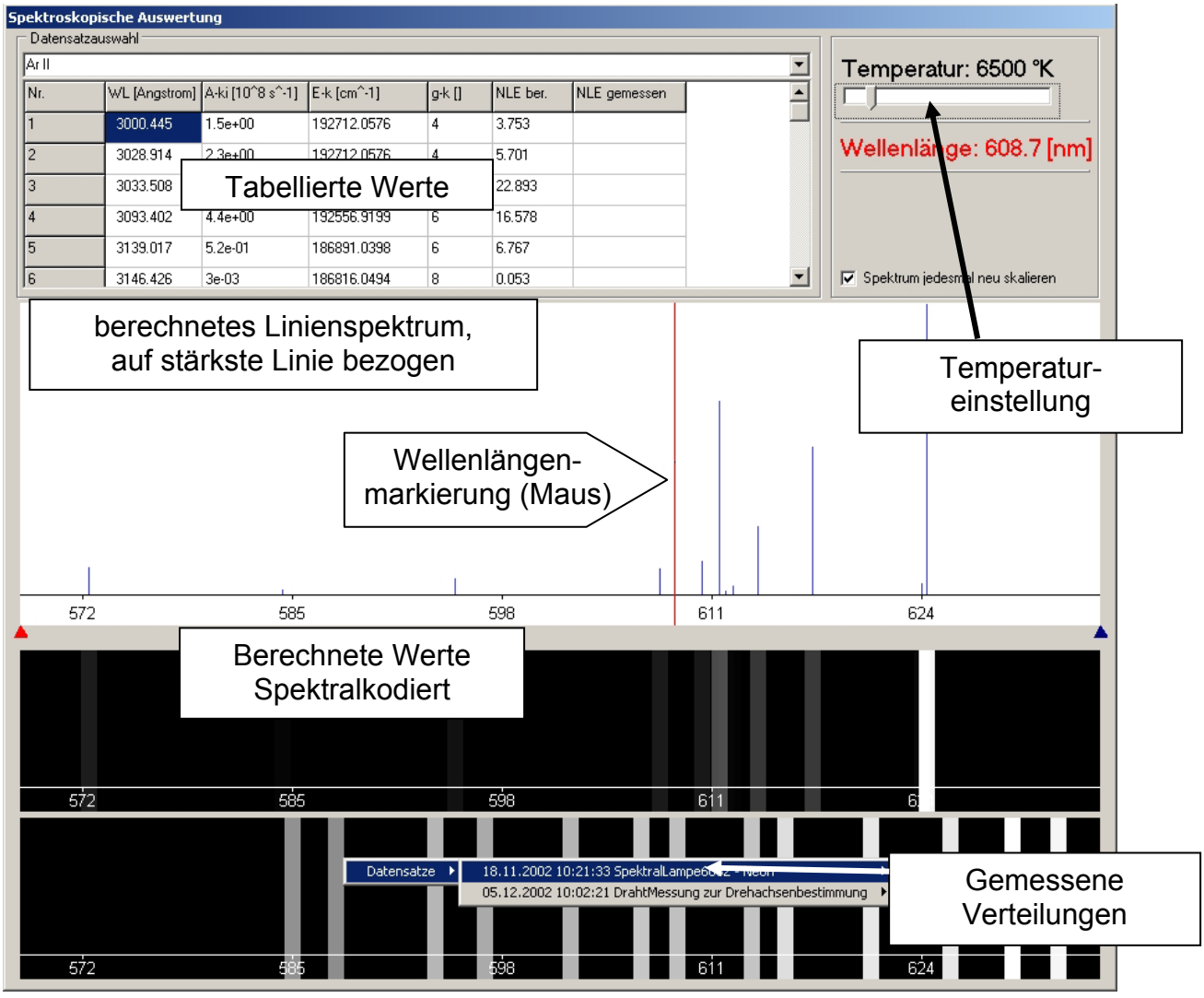


Abbildung 5-35: Formular Spektroskopische Auswertung

Mit Hilfe der Spektral-Justage haben wir die Möglichkeit, die Plasmatemperatur des aufgenommenen Wellenlängenbereiches zu bestimmen, indem mit diesem Formular die aufgenommenen Werte ihrer Intensität nach mit den zu erwartenden tabellierten Werten verglichen werden. Dazu werden je nach eingestellter Temperatur die verschiedenen Argon 1 oder Argon 2 Linien als Liniendiagramm in die Zeichnung eingetragen. Sie werden im unteren Bildbereich in helligkeitsabgestufte Spektralbalken

aufgetragen. Diese können dann mit den Messwerten verglichen werden, die ebenfalls am unteren Bildrand aufgetragen sind. Um eine Vergleichbarkeit zu gewährleisten wird dabei automatisch die linke und rechte Grenze des Anzeigenbereiches auf die vor der Messung eingestellten Randwerte gebracht. Durch Variation der Temperatur verändert sich die Intensität der einzelnen Linien. Auf diese Weise kann die Messung mit den unterschiedlichen berechneten Werten verglichen werden und kann eine erste Aussage über die Temperatur des Plasmas getroffen werden.

Die verwendeten Argonlinien werden aus einer im Internet verfügbaren Datenbank [14] entnommen. Um sie später schneller und einfacher verwalten zu können werden diese Daten mit Hilfe eines einfachen Konvertierungsprogramms in eine XML - Datei umgewandelt.

```
<Daten Gastype="Ar I">
…
   <Werte Spec="Ar I" WaveLength="3406.180" RelInt="" A_ki="3.9e-03"
    Acc="D" E_i="95399.8276" E_k="124749.805" Configurations="3s2.3p5(2P*<1/2>)4s-3s2.3p5(2P*<1/2>)7p" Terms="2[1/2]*-2[1/2]" J_i="1" J_k="0" g_i="3" g_k="1" _Type="" TP_Refs="8n"
    Line_Refs="" />
…
</Daten>
```

Abbildung 5-36: Datenbankstruktur Argon I (vgl. Hauptdatenbank)

Das Formular berechnet alle notwendigen Daten und stellt dies normiert dar. Auf diese Weise müssen die in Gleichung (9) enthaltenen Konstanten nicht berechnet werden, da sie durch die Normierung nicht mehr ins Gewicht fallen. Wellenlängenrichtig werden die errechneten Linien ähnlich den aufgenommenen Spektrallinien eingezeichnet. Dabei dient die Stärke der Linie als Umrechnungsfaktor für die Helligkeit der Spektrallinie. Verbreiternde Faktoren werden dabei nicht berücksichtigt. Die Linienbreite wird der geladenen Messung angepasst. Eine Normierung erfolgt wieder auf die „hellste“ Linie.
Die Messung kann in einem PopUpMenu ausgewählt werden. Dabei ist nur die Motorposition von Interesse, da zur Darstellung alle aufgenommenen Linien notwendig sind. Diese werden ähnlich wie die berechneten Spektren aufgetragen. Dazu werden alle gespeicherten aufgenommenen Pixelwerte addiert.

Zu beachten ist hier, die berechneten scharfen Spektrallinien müssen als Band dargestellt werden. Dadurch entstehen Überlagerungen von Bändern, die als dünne, sehr helle Streifen zu sehen sind.
Zum jetzigen Zeitpunkt ist es nicht möglich die Plasmatemperatur direkt abzulesen. Es besteht lediglich die Möglichkeit die aufgenommenen Spektrallinien mit den tabellierten Werten von Hand zu vergleichen.

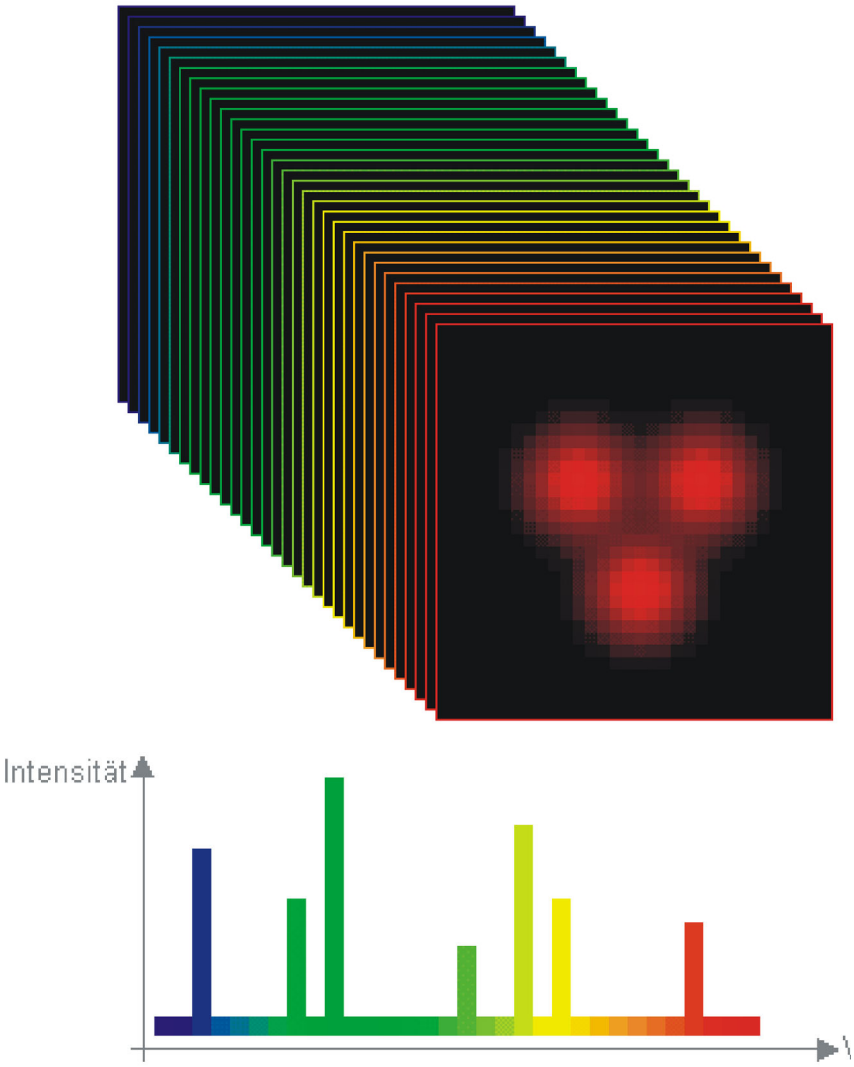


Abbildung 5-37: Datenauswertung

5.5.5 Das Formular „3D Browser“

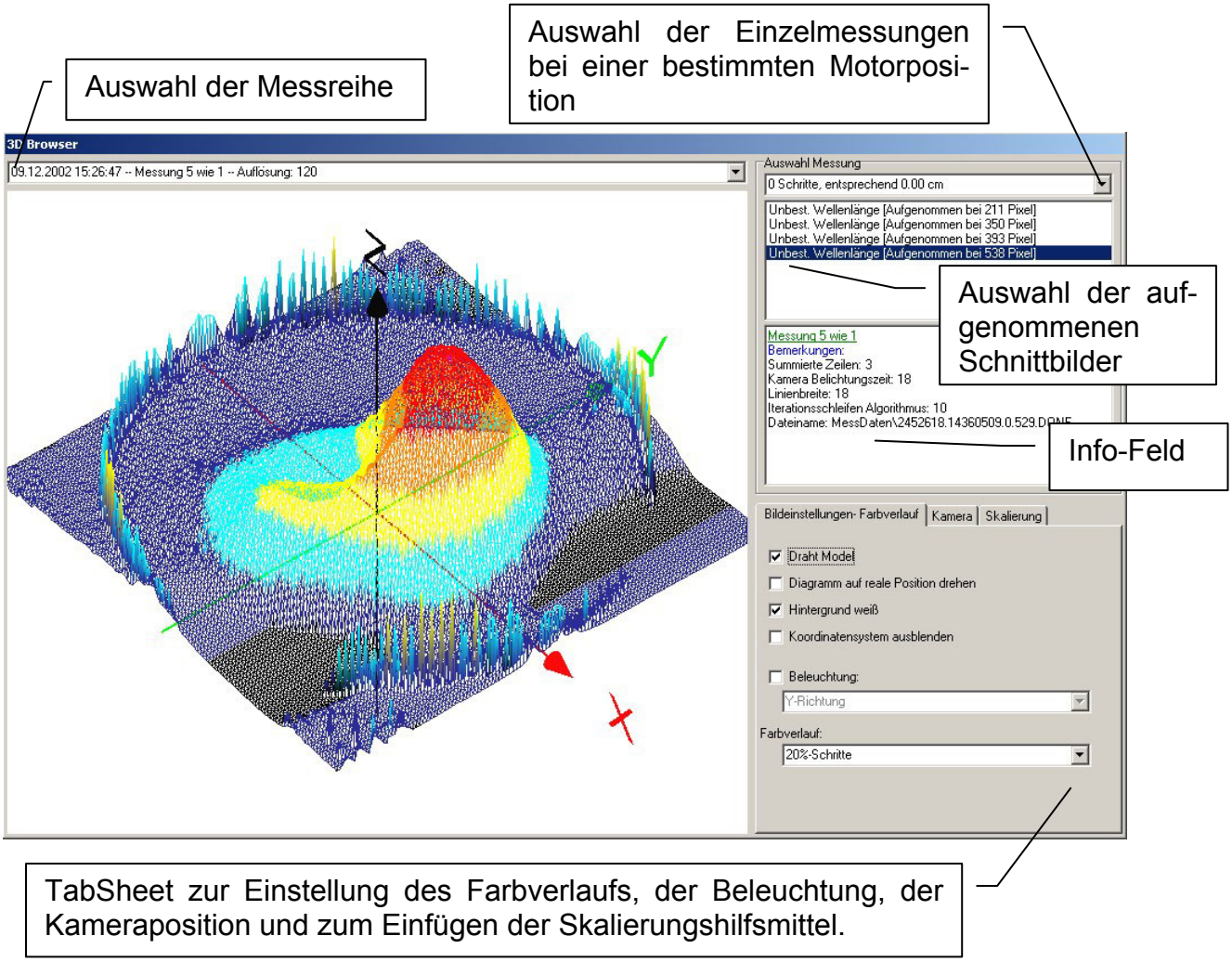


Abbildung 5-38: Das Formular "3D Browser"

Beschreibung

Das Formular „3D Browser“ dient der grafischen Veranschaulichung der vom Tomografiealgorithmus ausgewerteten Messdaten.

Dabei ist zu beachten, dass nicht die reelle Ausdehnung des Messobjektes, sondern die Intensität der Emission **einer Wellenlänge** bei **einem bestimmten Schnitt** dargestellt wird.

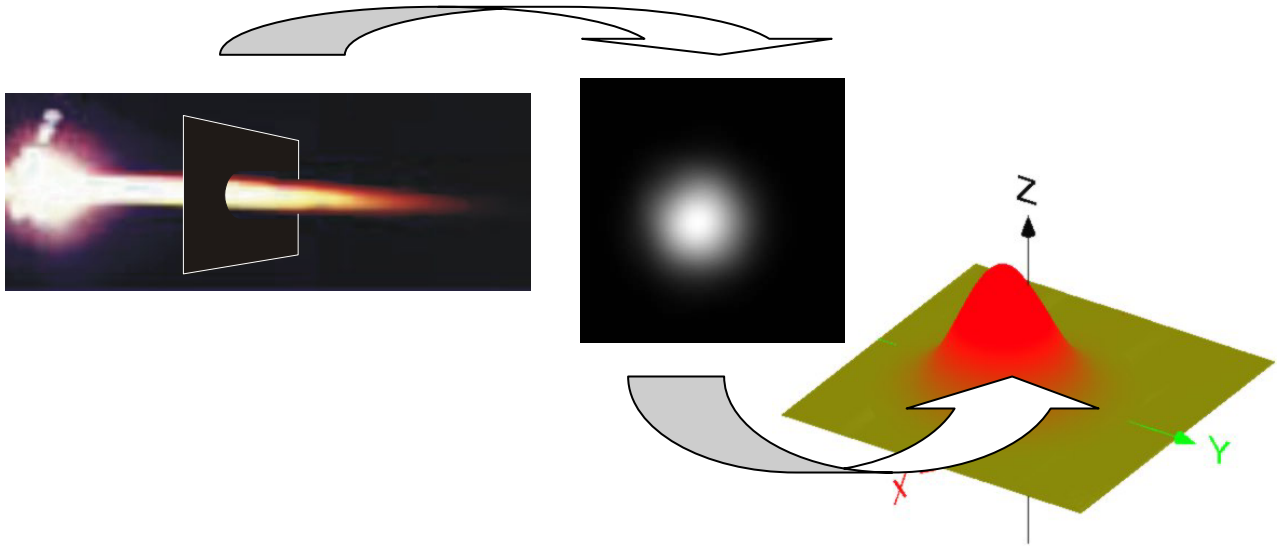


Abbildung 5-39: Zustandekommen der Messwerte

Weiterhin sind die Messwerte auf eins normiert. Sie können bezüglich der Höhe (also der Strahlungsintensität) nicht mit anderen Messungen verglichen werden.
Dafür werden minimale Nuancen, die im 2D-Bild nicht ausgemacht werden können, sichtbar.
Die folgenden beiden Bilder stammen aus einer Testmessung am 5.12.02, bei der zusätzlich zum eigentlichen Trägergas Kaltgas zwischen den Kathoden einströmte.

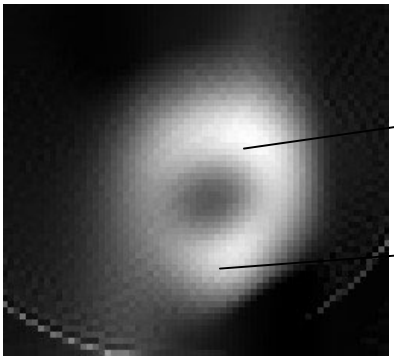


Abbildung 5-40: Ausgewertetes Bild

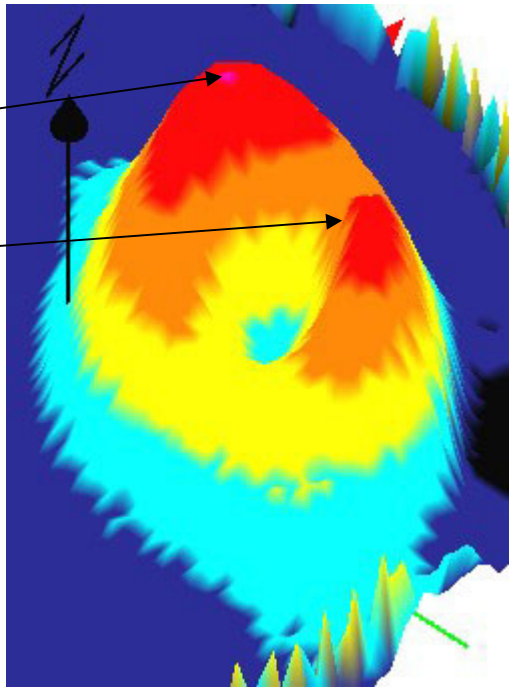


Abbildung 5-41: 3D-Darstellung eines ausgewerteten Bildes

Die durch das Kaltgas verursachte „kalte Seele“ ist hier gut zu sehen.
Besonders deutlich ist auf dem 3D-Bild zu erkennen, dass nur zwei Maxima an Stelle der erwarteten drei auftreten.
Eine Erklärung, woher dieser Effekt kommt, könnte sein, dass die Kathoden des Brenners unterschiedlich stark verschlissen sind und daher die Bogenströme voneinander abweichen.

Zu den verschiedenen Einstellungsmöglichkeiten:

Bildeinstellungen und Farbverlauf:

Draht-Model: Das HeightField wird als Drahtmodel dargestellt und somit transparent.

Diagramm auf reale Position drehen:
Die Koordinatenachsen werden um 115° gedreht. Dies ist der Winkel zwischen dem realen Koordinatensystem und dem Koordinatensystem des aufgenommenen Bildes.

Hintergrund weiß:
Der Hintergrund des OpenGLViewers wird weiß dargestellt, was praktisch ist, möchte man einen ScreenShot ausdrucken.

Koordinatensystem ausblenden:
Ebenfalls zum Ausdrucken eines ScreenShots: Sollte das Koordinatensystem dabei stören, so kann es ausgeblendet werden.

Beleuchtung: Hiermit kann die Position der Beleuchtung relativ zum Koordinatensystem verändert werden. Dies kann vorteilhaft sein, falls man bestimmte Bildbereiche hervorheben möchte.

Farbverlauf: Zur Skalierung bietet sich hier der Verlauf 20% an, bei dem der Graph in 5 Höhenbereiche unterteilt und in verschiedenen Farben dargestellt wird. Der höchste Punkt (die Stelle der stärksten Emission) ist hellrot markiert.

Kamera:
Bestimmt die Position der Kamera relativ zum Koordinatensystem. Es kann somit von verschiedenen Perspektiven angesehen werden.

Skalierung:
Ein Raster (Einteilung 25) erleichtert das genaue Vergleichen einzelner Messwerte **einer** Messung.

Realisierung im Quelltext:

Das zentrale Element im GL-Browser ist das sogenannte Heightfield. Es ist ein Objekt in OpenGL zum Darstellen dreidimensionaler Flächen und eignet sich insbesondere zum Visualisieren einer Funktion im kartesischen Koordinatensystem.
Die Übergabe einzelner Funktionswerte gestaltet sich beim Heightfield recht aufwendig: Zunächst muss die „Auflösung“ in den Variablen `XSamplingScale.Step` und `YSamplingScale.Step` eingestellt werden. Sie speichern die Schrittweite, in der vom Minimalwert (hier 0) zum Maximalwert (hier 5) gesprungen wird. Angenommen die Auflösung ist 60, so ist die Schrittweite (5-0)/(60-1)=0,085.
Ausgehend von diesen Informationen wird im Anschluss die Prozedur

```
MessBild(const x, y: Single; var z: Single;
          var color: TColorVector; var texPoint: TTexPoint)
```

vom HeightField so lange aufgerufen, bis alle erwarteten Funktionswerte (Anzahl Auflösung^2) abgearbeitet sind. Ausschlaggebend hierfür ist die Zeile `HeightField.OnGetHeight:=MessBild` in `TGLBrowser_Form.BildAufbauen()`.
Der Prozedur `MessBild` werden einige Konstanten und Variablen übergeben:
Die Konstanten x und y werden, ausgehend von 0, um jeweils die Schrittweite hochgesetzt, woraus die Variable `count` errechnet wird, welche die Position des gerade aktiven Pixels im MemoryStream speichert. Ausgehend von dem Wert des Eintrages im MemoryStream wird z, auf die maximale Höhe der z-Achse normiert, gesetzt.

TColorVektor speichert die Farbe des jeweiligen Pixels, die je nach eingestelltem Farbverlauf von der z-Koordinate abhängt. TColorVektor besteht aus der reinen Farbinformation Rot-Grün-Blau, erweitert um einen Wert für die Transparenz.
Eine Prozedur, die sehr komfortabel einen Farbverlauf erzeugt, ist
`VectorLerp(clrGreen, clrRed, z, color)` (hier von grün nach rot, abhängig von z).
Da keine Textur verwendet wird, kann die Variable texPoint unverändert bleiben.
Damit nun auf dem Bildschirm keine unebene, pixelige Fläche entsteht, werden intern mehrere Werte zwischen den eigentlichen Funktionswerten interpoliert. Man erhält somit eine ebenmäßige, glatte Oberfläche.

5.6 Zusätzliche Hilftsformulare

5.6.1 Das Formular „Messdatenaufarbeitung“

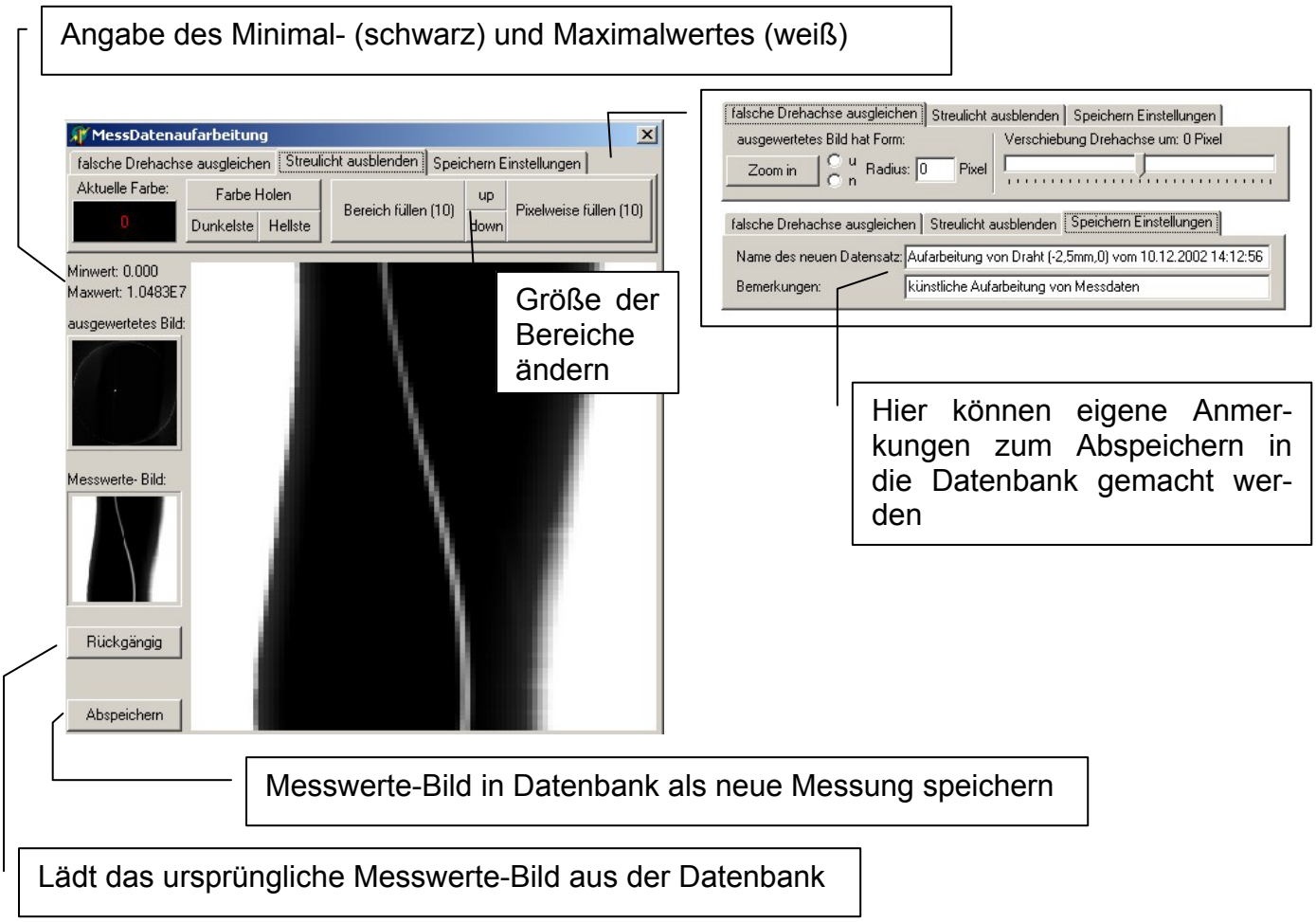


Abbildung 5-42: Das Formular "MessDatenAufarbeitung"

Beschreibung

Gelegentlich treten aufgrund äußerer Einflüsse bei Messungen Fehler auf. Gerade die Einstellung der Drehachse birgt einige Fehlerquellen: Angefangen bei der Justage bis hin zu den eigentlichen Messungen.

Ein sehr unangenehmer Fehler ist das „Störlicht“, das bei der Drehachsjustage auftritt: Ist die Leuchtkraft der Glühbirne für die Hinterleuchtung des Drahtes zu schwach, so entstehen dunkle bis schwarze Ränder links und rechts am Bildrand. Wird dieses Bild nun invertiert, entstehen helle Ränder (siehe oben). Die Auswertung eines solchen Bildes mit dem Tomoalgorithmus kann nicht genau sein, da zumeist die Fehler (großer Ring im ausgewerteten Bild) größer sind als die wichtigen Messwerte.

Ein weiterer Fehler ist die falsch eingestellte Drehachse, der eine horizontale Verschiebung der Messwerte bewirkt (siehe „Das Formular Drehachsenbestimmung“).

Wie kann man nun Messdaten aufbereiten?

Sollte einer der beiden oben genannten Fehler auftreten, kann das Formular „Messdatenaufarbeitung“ im Formular „Tomografische Auswertung“ durch Rechtsklick auf die fehlerhafte Messung gestartet werden.

Nun gibt es zwei Möglichkeiten:

a) Fehler durch falsch eingestellte Drehachse beheben

Dies ist der am leichtesten zu behebende Fehler: Wie bei der Drehachsjustage muss zunächst das ausgewertete Bild herangezoomt werden. Danach ist die Form der sichelförmigen Gestalt auszuwählen, deren Radius einzugeben und mit Return zu bestätigen.

b) Fehler durch „Streulicht“ beheben

Hier müssen die deutlich sichtbaren hellen Ränder des Messwerte-Bildes per Hand schwarz gefüllt werden, wobei die eigentlichen Messwerte auf gar keinen Fall verändert werden dürfen, da sonst eine Fehlerquelle durch eine andere, folgenschwerere, ersetzt wird.

Im Normalfall benötigt man die dunkelste Farbe, die defaultmäßig eingestellt ist. Zur Auswahl einer anderen Farbe einfach den Button Farbe holen betätigen und durch Anklicken auf dem Image auswählen.

An den Buttons Pixelweise füllen (quadratischer Bereich) und Bereich füllen (Bereich, der sich über die gesamte Höhe des Bildes erstreckt) stellt man die Art des Pinsels ein.

Vergleich der Messdaten und deren Auswertungen vorher und nachher

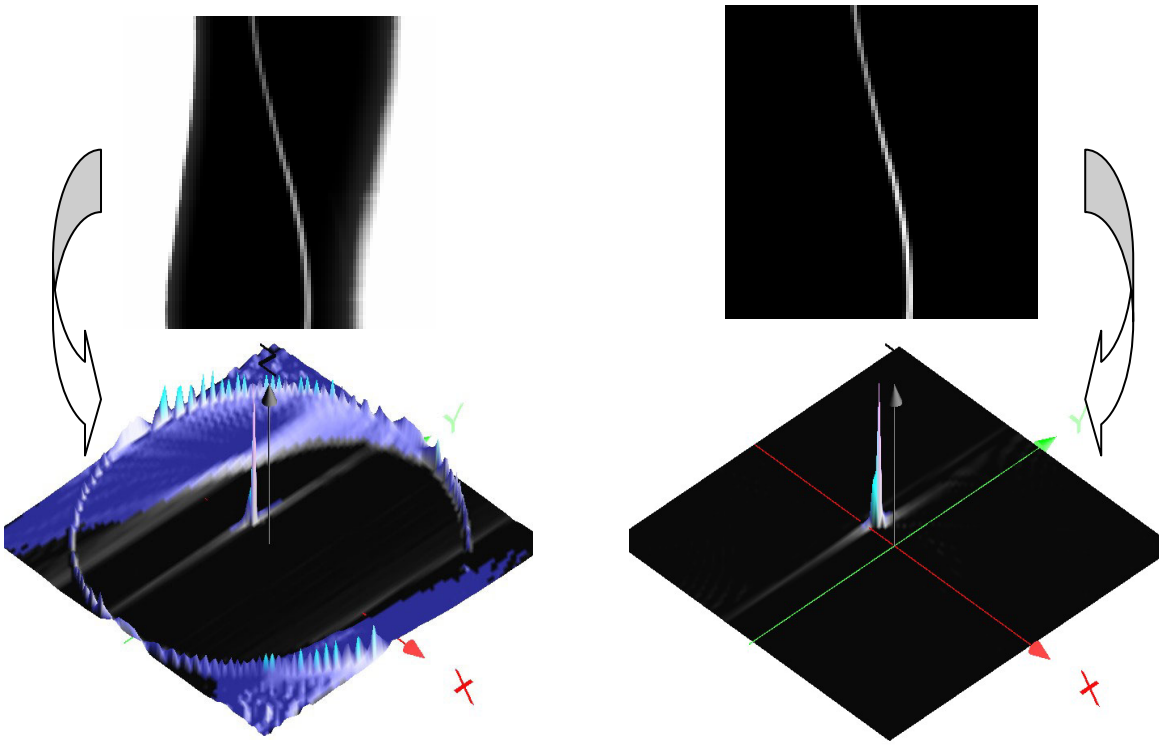


Abbildung 5-43: Messdaten vor und nach der Aufbereitung

Realisierung im Quelltext

Vom Formular „Tomografische Auswertung“ werden die beiden Integer-Variablen `Datensatz` und `Einzelmessung` übergeben. Bei `TAufarb_Form.FormShow` wird mit ihrer Hilfe das zugehörige Messwerte- und -wenn vorhanden- das Ausgewertete Bild geladen. Weiterhin werden Maximalwert (weiß) und Minimalwert (schwarz) ermittelt.

Zum horizontalen Verschieben des Messwerte-Bildes dient die Prozedur `TAufarb_Form.StreamVerschieben(Wert: Integer)`. Sie erzeugt zunächst eine Sicherungskopie des MemoryStream, in dem die Bilddaten gespeichert sind, und verschiebt ihn dann um die in Wert angegebene Anzahl von Pixeln. Dies geht folgendermaßen vonstatten: Mit einer for-Schleife wird der Stream durchlaufen, aus der Laufvariablen i dann Zeile und Spalte des gerade betrachteten Pixels ermittelt und vom gesicherten Stream der Farbwert in den neuen Stream an der um Wert verschobenen Position eingefügt. Aus dem Bereich herausgeschobene Pixel verfallen, die auf der anderen Seite neu eingefügten Pixel erhalten die Farbe schwarz.
Man könnte diese Aufgabe auch mit nur einem Stream lösen, jedoch wäre dann nur die Verschiebung in eine Richtung möglich, beim Zurückschieben würden Informationen verloren gehen. Ich entschied mich deshalb für die einfachere, allerdings speicheraufwendigere Version.

Die Aufgabe des Füllens bestimmter Bereiche im Messwerte-Bild übernehmen die Prozeduren `TAufarb_Form.MalePixel(X,Y: Integer)` und `TAufarb_Form.FuelleBereich(links, rechts: Integer)`.
Mit `TAufarb_Form.BildAktualisieren()` wird dann das veränderte Bild auf dem Bildschirm ausgegeben.

Zuletzt soll nun noch betrachtet werden, wie das Heranzoomen des ausgewerteten Bildes realisiert ist:
Nachdem der Button Zoom in betätigt wurde, wird die Statusvariable `Zustand` auf `zoom` gesetzt. Mit Linksklick auf `Image1` gibt der Benutzer den zu vergrößernden Bereich, charakterisiert durch dessen Mittelpunkt, an. Prozedur `TAufarb_Form.ZoomImage(X: Integer; Y: Integer)` führt nun die Vergrößerung durch. Sie durchläuft das ursprüngliche Bitmap nach Zeile und Spalte und belegt jeweils 12 Pixel mit dem Wert eines Pixels aus dem zu vergrößernden Bereich (`Wert:=tmp[3*Trunc(links+j/12)];` tmp: Array aus allen Pixeln einer Zeile des Ursprungsbildes, links: Variable, die die Spaltennummer enthält, die den zu vergrößernden Bereich nach links begrenzt).

5.6.2 Das Formular „Checklist“

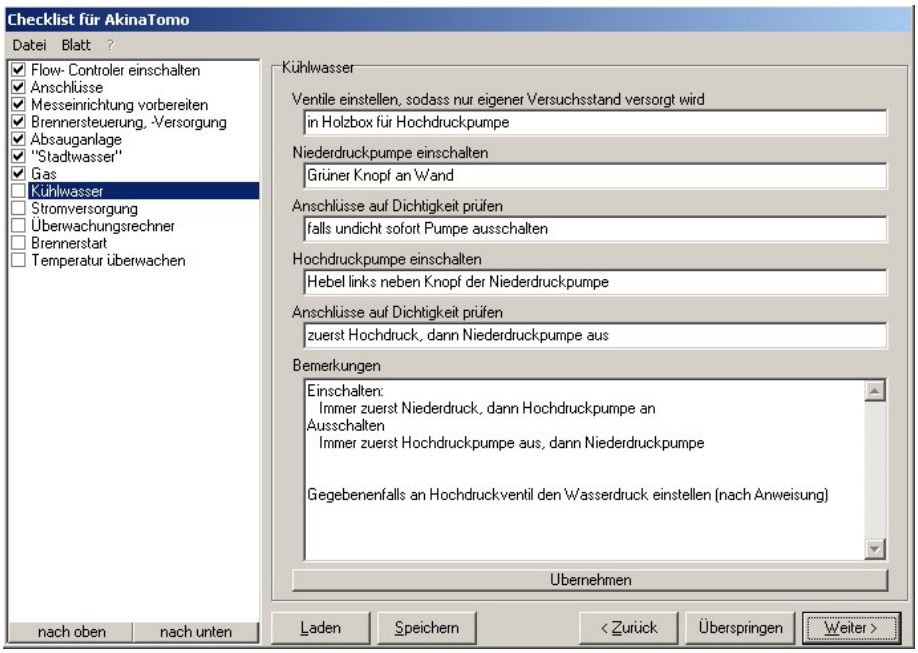


Abbildung 5-44: Das Formular "Checklist"

Beschreibung:

Das Programm Checklist hat im eigentlichen Sinne wenig mit der tomografischen Emissionsspektroskopie zu tun. Es ist vielmehr ein Hilfsmittel, um Vorgänge und Arbeitsanweisungen darstellen und speichern zu können. Es wurde entwickelt, um jedem Benutzer der Akina-Software einen Überblick über Vorgehensweisen, sei es die optische Justage, die Inbetriebnahme des Brenners etc. zu geben.
Das Programm besteht im Wesentlichen aus einer Liste einzelner Blättern, die durch Anklicken betrachtet und bearbeitet werden können. Mit den Buttons Weiter und Zurück springt man vorwärts bzw. rückwärts durch die einzelnen Blätter, die je nachdem als erledigt abgehakt werden.

Erstellen von Checklistblättern:

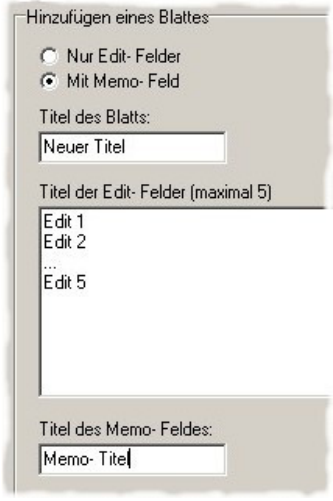


Abbildung 5-45: Erstellen eines Blattes

Dieser Ausschnitt zeigt einen Teil der GroupBox zum Hinzufügen eines neuen Blattes. Vor dem Erstellen muss entschieden werden, ob ein Memo-Feld und wie viele Edit-Felder benötigt werden. Deren Titel wird hier bereits festgelegt, er kann später nicht mehr editiert werden.
Die maximale Anzahl der Edit-Felder hängt von der Entscheidung für ein Memo-Feld ab. Ohne dieses passen neun Edit-Felder auf ein Blatt - mit nur fünf!
Der Benutzer beendet das Erstellen mit einem Click auf OK, woraufhin das Blatt mit seinem Titel in die Checklist übernommen wird. Von dort kann es aufgerufen und dann mit Information gefüllt werden.

Das zu dem obigen Beispiel gehörige Blatt sähe dann so aus:

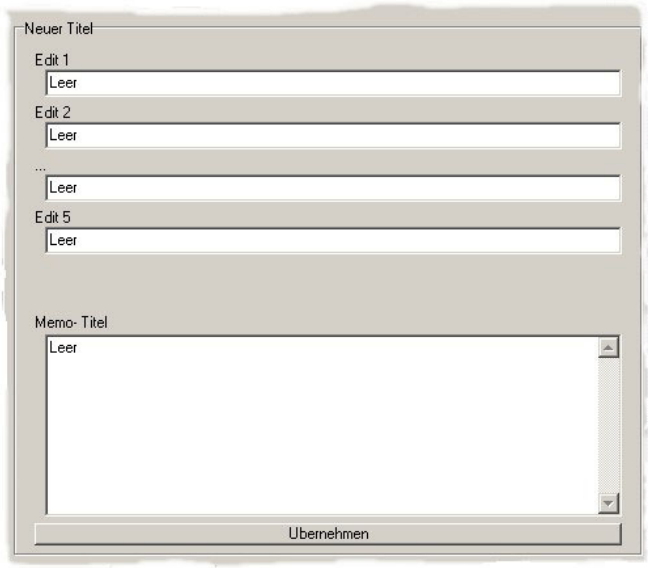


Abbildung 5-46: Beispielblatt

Zur Realisierung im Quelltext:
Checklist ist ein eigenständiges Programm, da es auch unabhängig von der Akina-Software lauffähig sein soll. Der Nachteil ist jedoch, dass es dann nicht direkt aus dem Hauptformular der Akina-Software beeinflusst werden kann. Vielmehr müssen Windows-Botschaften als Informationsträger zwischen den Programmen dienen:
Grundsätzlich ist eine Windows-Botschaft vom Typ TMessage.

Auszug aus der Delphi-Hilfe:

```
type
   TMessage =packedrecord
    Msg: Cardinal;
    caseIntegerof
      0: (WParam: Longint; LParam: Longint; Result: Longint);
      1: (WParamLo: Word; WParamHi: Word; LParamLo: Word; LParamHi: Word; ResultLo: Word;
        ResultHi: Word);

  end;
```

Uns interessiert hauptsächlich die Variable Msg, in der die eigentliche Botschaft in Form einer Zahl versteckt ist. Alle Zahlen größer 1024 können verwandt werden, die Zahlen darunter sind von Windows reserviert. Um eine Botschaft empfangen zu können muss die Standart- Botschafts- Behandlungsroutine
`WndProc(var Message: TMessage)` um die eigenen Behandlungsroutinen ergänzt werden (Vorsicht, da im Programmablauf mehrere hundert Botschaften pro Sekunde denkbar wären, könnte dies die Ausführung des Programms stark verlangsamen).
Das Versenden einer Botschaft übernimmt das Windows Messaging-Konstrukt `PostMessage(hWnd: HWND; Msg: Cardinal; wParam: Integer; lParam: Integer)` Allerdings wird hier das Handle auf die Anwendung, die die Botschaft empfangen soll benötigt. In diesem Fall erhält man das Handle einfach über `FindWindow(lpClassName: PChar; lpWindowName: PChar)`, da der Titel der Anwendung (die Caption der Form) bekannt ist.
In der Mainframe- Form wird das Ausführen der Checklist.exe von einem `ActionManager` (`FileRun`) übernommen. Zusätzlich werden noch Parameter übergeben, sodass bei einem Öffnen der Checklist aus der Akina-Software automa-

tisch die BorderIcons verschwinden und die Standart-Caption „Checklist für Akina-Tomo“ erscheint (Dies ist wichtig, um leicht das zugehörige Fenster-Handle zu erhalten).
Nach dem Öffnen wird das Checklist-Fenster entweder mit `Windows.BringWindowToTop(W)` in den Vordergrund gebracht oder mit `PostMessage(W, wm_close, 1, 1)` geschlossen (`W: HWND; wm_close = 16`).
Das Beenden der Akina-Software schließt automatisch auch die geöffnete Checklist!

Die einzelnen Blätter sind intern als `TBlatt` in einer Liste `BlattListe` gespeichert:

Auszug aus `ChecklistUnit`:

```
TBlatt = class
         Titel: String;
         Edits: TList;
         Memo: Boolean;
         MemoText: TStrings;
         MemoTitel: String;
         Status: TStatus;
end;

TStatus = (erledigt, uebersprungen, nichterledigt);
```

Da jedes Blatt eine unterschiedliche Anzahl an Edit-Feldern umfasst, sind diese wiederum in einer Liste `Edits` als `TTEdit` abgelegt:

```
TTEdit = Class
         Inhalt: WideString;
         Titel: String;
end;
```

Die GroupBox `HinzuBox` ist bereits zu Programmbeginn initialisiert und wird bei Bedarf entweder mit `Show` gezeigt oder mit `Hide` versteckt. Die `BlattBox` hingegen wird erst während der Laufzeit mit Komponenten gefüllt (siehe hierzu die Prozedur `ZeigeBlatt(Nummer: Integer)`).

Abschließend möchte ich noch auf die Speicherung der Blattliste in eine *.Checklist - Datei eingehen:
Das Problem hierbei ist die unterschiedliche Länge und Anzahl der einzelnen Strings in den jeweiligen Blättern. Somit müssen im Bitstream der Datei zusätzliche Verwaltungsinformationen über die Länge oder Position der einzelnen Strings gespeichert werden.

Ich wählte die folgende Struktur:

Kopfteil:

Beschreibung	Datentyp	Anzahl belegter Bytes
Anzahl Blätter	Integer	4
Beginn erstes Blatt	LongInt	4

je Blatt:

	Beschreibung	Datentyp	Anzahl belegter Bytes
	Länge Titel	Integer	4
	Titel	String	Länge Titel
	Anzahl Edits	Integer	4
	Edits	TTEdit	s. Editbeschreibung
	Memo vorhanden?	Boolean	4 (als Int gespeichert)
optional	Länge Titel	Integer	4
optional	Titel	String	Länge Titel
optional	Anzahl der Zeilen	Integer	4
optional	Länge Zeile 1	Integer	4
optional	Zeile 1	String	Länge Zeile 1
optional	…	…	…
optional	Länge Zeile n	Integer	4
optional	Zeile n	String	Länge Zeile n

je Edit:

Beschreibung	Datentyp	Anzahl belegter Bytes
Länge Titel	Integer	4
Titel	String	Länge Titel
Länge Inhalt	Integer	4
Inhalt	String	Länge Inhalt

Realisiert ist dieses Schema in den Prozeduren `ExpToStream(Pfad: WideString)` und `StreamToExp(Pfad: WideString)`, die nach Aufruf sofort den Inhalt einer Blattliste in einen mit `Pfad` bestimmten FileStream schreibt bzw. aus ihm liest.
Beispiel: Die Checklist, bestehend aus einem Blatt „Name“ mit einem Edit-Feld „Edit“ und dem Memo-Feld „Memo“ (beide Felder mit dem String „Leer“ gefüllt), hätte im gespeicherten Zustand eine Länge von 60 Bytes:

Anzahl Blätter	(=1)	4 Byte
Anfang erstes Blatt	(=8)	4 Byte
Länge Titel Blatt1	(=4)	4 Byte
Titel Blatt1	(=’Name’)	4 Byte
Anzahl Edits	(=1)	4 Byte
Länge Titel Edit1	(=4)	4 Byte
Titel Edit1	(=’Edit’)	4 Byte
Länge Inhalt Edit1	(=4)	4 Byte
Inhalt Edit1	(=’Leer’)	4 Byte
Memo da ?	(=True)	4 Byte
Länge Titel Memo	(=4)	4 Byte
Titel Memo	(=’Memo’)	4 Byte
Anzahl Zeilen	(=1)	4 Byte
Länge 1.Zeile	(=4)	4 Byte
Inhalt 1.Zeile	(=’Leer’)	4 Byte
		=60 Byte

5.6.3 Das Formular „Boltzmann – Plot“

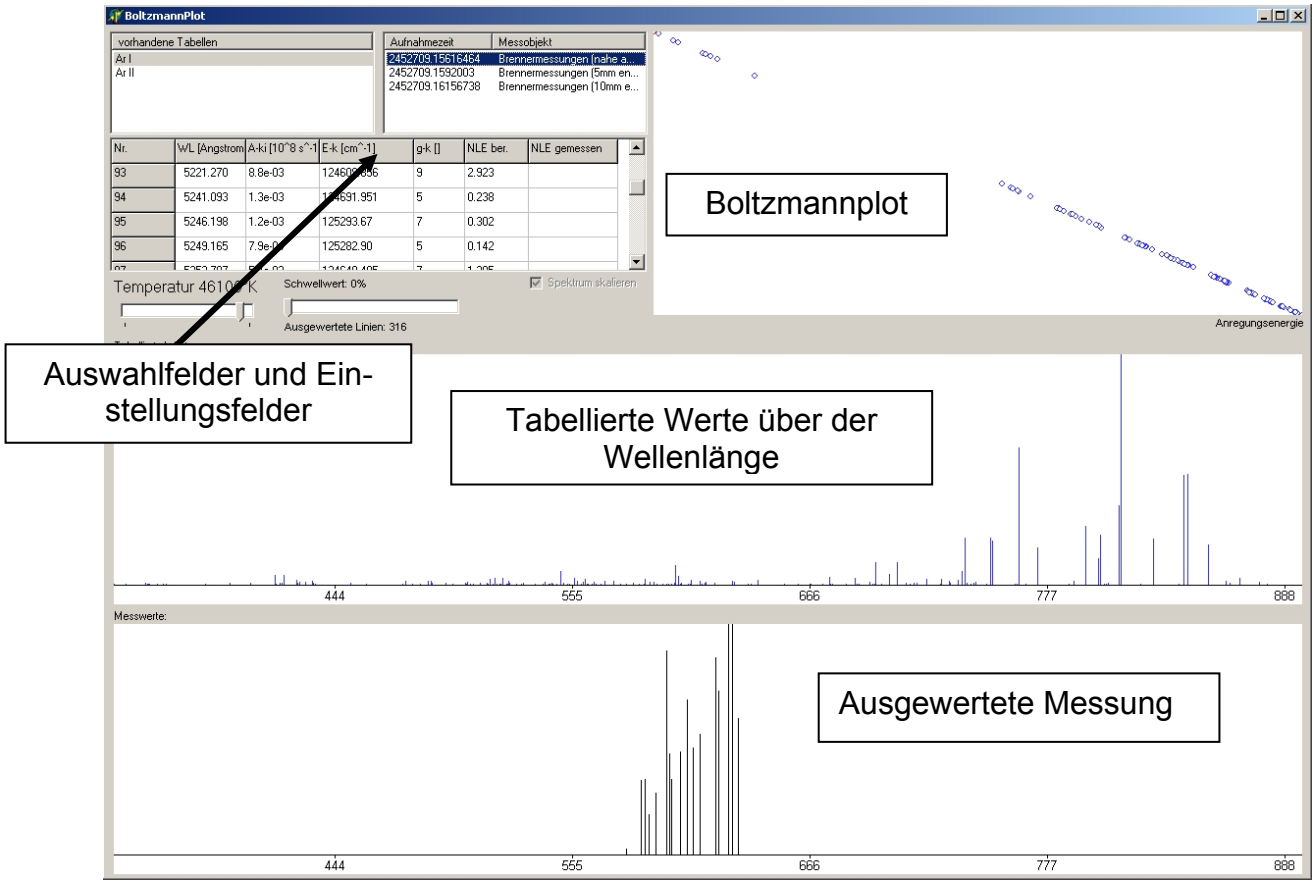


Abbildung 5-47: Boltzmannplot

Hier handelt es sich, ähnlich dem Formular Spektroskopische Auswertung, um die Betrachtung einer ganzen Anzahl von Spektralbereichen auf einmal um eine Aussage über die tatsächliche Temperatur des Plasmas zu erhalten.
In der Mitte des Formulars lassen sich die vorher im Einstellungsbereich ausgewählten Nettolinienemissionen darstellen. Grundlage der Berechnung sind auch hier die dem Programm zur Verfügung stehenden tabellierte Werte. Im untern Bild werden dann die vom Versuchsaufbau aufgenommenen Werte dargestellt. Es handelt sich auch bei vollständig aufgezeichneter Messung nur um die Linien, die in der Datenbank gespeichert sind. Beide Bilder sind auf die jeweils stärkste Linie normiert und über der Wellenlänge (in [nm]) aufgetragen.
Die wichtigste der hier sichtbaren Darstellungen ist allerdings die Darstellung in der rechten oberen Ecke.
Es handelt sich hier um die in Kapitel 0 (vgl. Gleichung (9)/ Abbildung 3-16) hergeleitete Beziehung. Bei tabellierten Werten (blau) wie hier ergibt sich eine Gerade. Auch bei Auswertung der Messwerte (rot) muss eine Gerade entstehen. Sollte dies nicht der Fall sein muss festgestellt werden welche Gründe (wahrscheinlich Auswerte- oder Messfehler) dies hat.

5.6.4 Das Formular „HTML – Konverter“

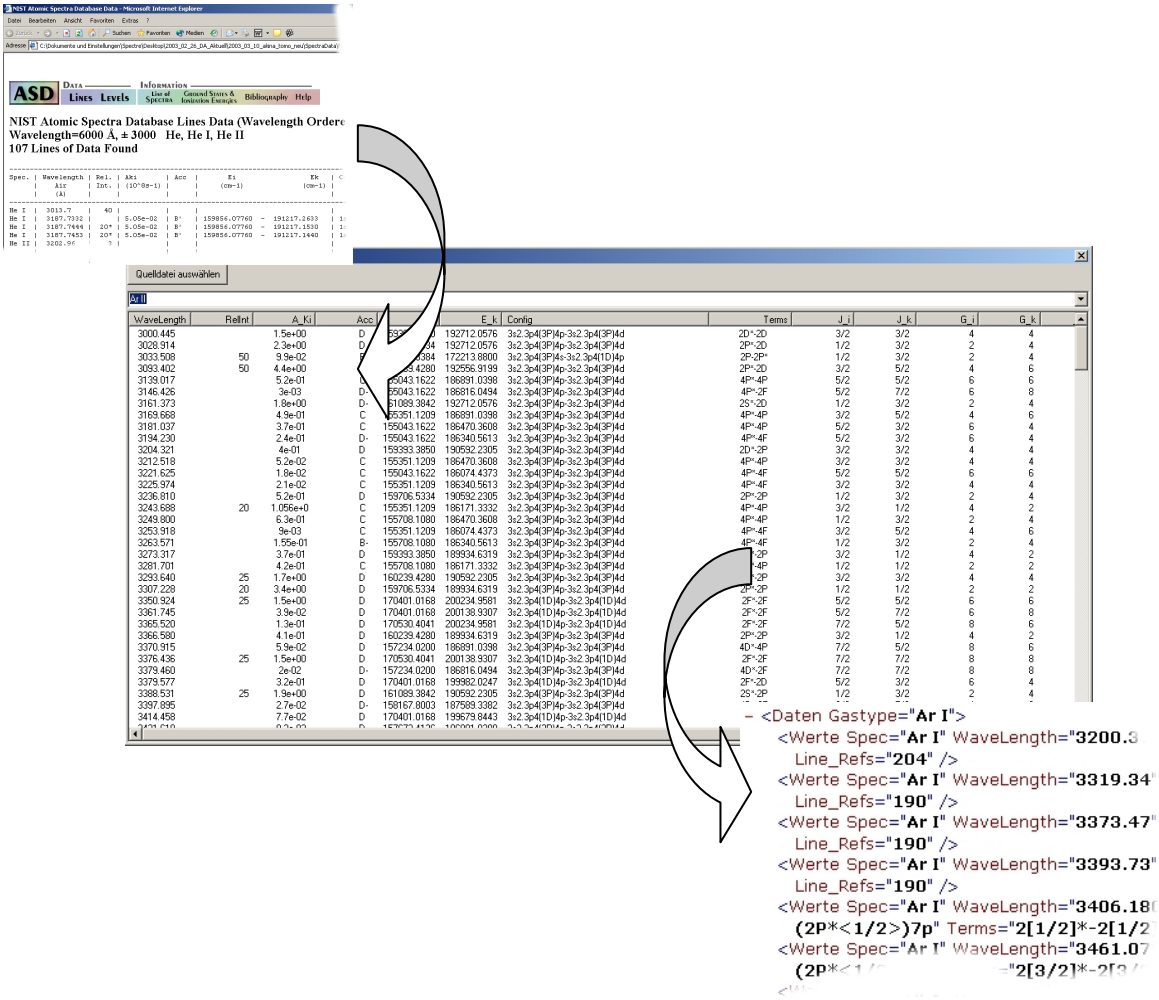


Abbildung 5-48: Das Formular HTML – Konverter

Die im Internet erhaltenen Daten können nicht direkt verarbeitet werden. Sie müssen lokal gespeichert werden. Die so erhaltenen HTML – Dateien können dann mit Hilfe des Formulars verarbeitet werden. Dabei wird nachdem die Datei geladen wurde (Schaltfläche „Quelldatei laden“) automatisch eine XML – Datei angelegt. Sie kann dann wieder ohne Probleme von anderen Formularen weiterverarbeitet werden. Eine optische Kontrolle kann im unteren Teil des Formulars stattfinden, ist aber nicht nötig. Die Umwandlung der HTML – Dateien geschieht dabei durch geschickte Zeilenzerlegung des HTML – Quellkodes. Eine Unit zum Zugriff der XML – Datei ist ebenfalls aus anderen Formularen bereits vorhanden und stellt somit keine Schwierigkeit dar.

5.6.5 Das Formular „Messwerte addieren“

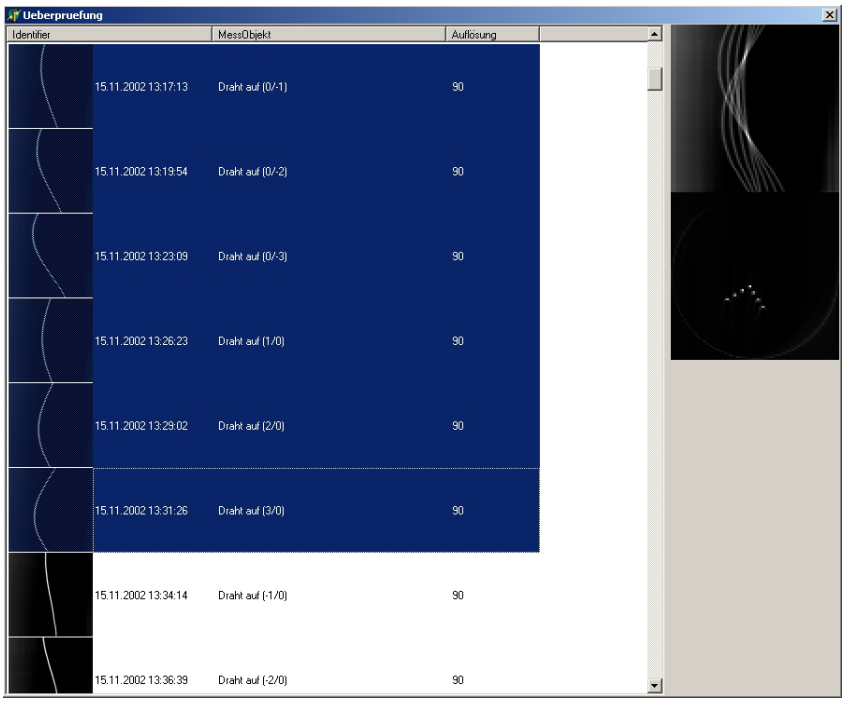


Abbildung 5-49: Formular „Messwerte addieren“

Um die Justierung der Messapparatur zu unterstützen wurde dieses Formular entwickelt. Durch Drahtmessungen kann die Drehachse festgestellt werden. Mehrere dieser Messungen lassen eine Aussage über Abbildungsmaßstab sowie Koordinatenachsendrehung der Optik zu (vgl.Abbildung 5-50). Eine Messung an mehreren gespannten Drähten ist nicht möglich. Eine Invertierung des Bildes hilft nicht, da überlagerte Bildbereiche („Aufnahme von mehreren Drähten übereinander“) nicht wie Lichtquellen doppelt so hell erstrahlen. Die so entstehenden Fehler können nur durch Summierung mehrerer Messreihen durchgeführt werden.

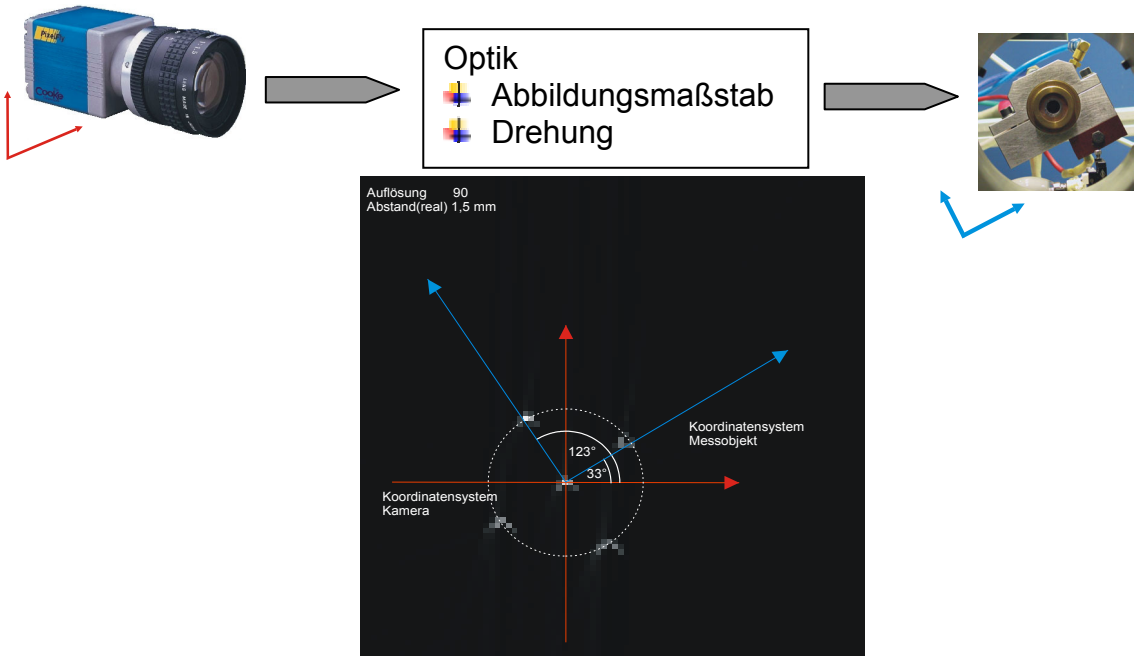


Abbildung 5-50: Prinzip Kontrolle der Koordinatensysteme

6 Messungen und Ergebnisse

Um die Funktionsfähigkeit der Messapparatur und der dazu gehörigen Software tomografisch wie spektorskopisch zu testen wurden mehrere Testmessungen vorgenommen.
Zuerst allerdings soll festgestellt erden wie sich die Koordinatensysteme der Apparatur und des Spektrografen unterscheiden. Dazu dient folgender Messaufbau:

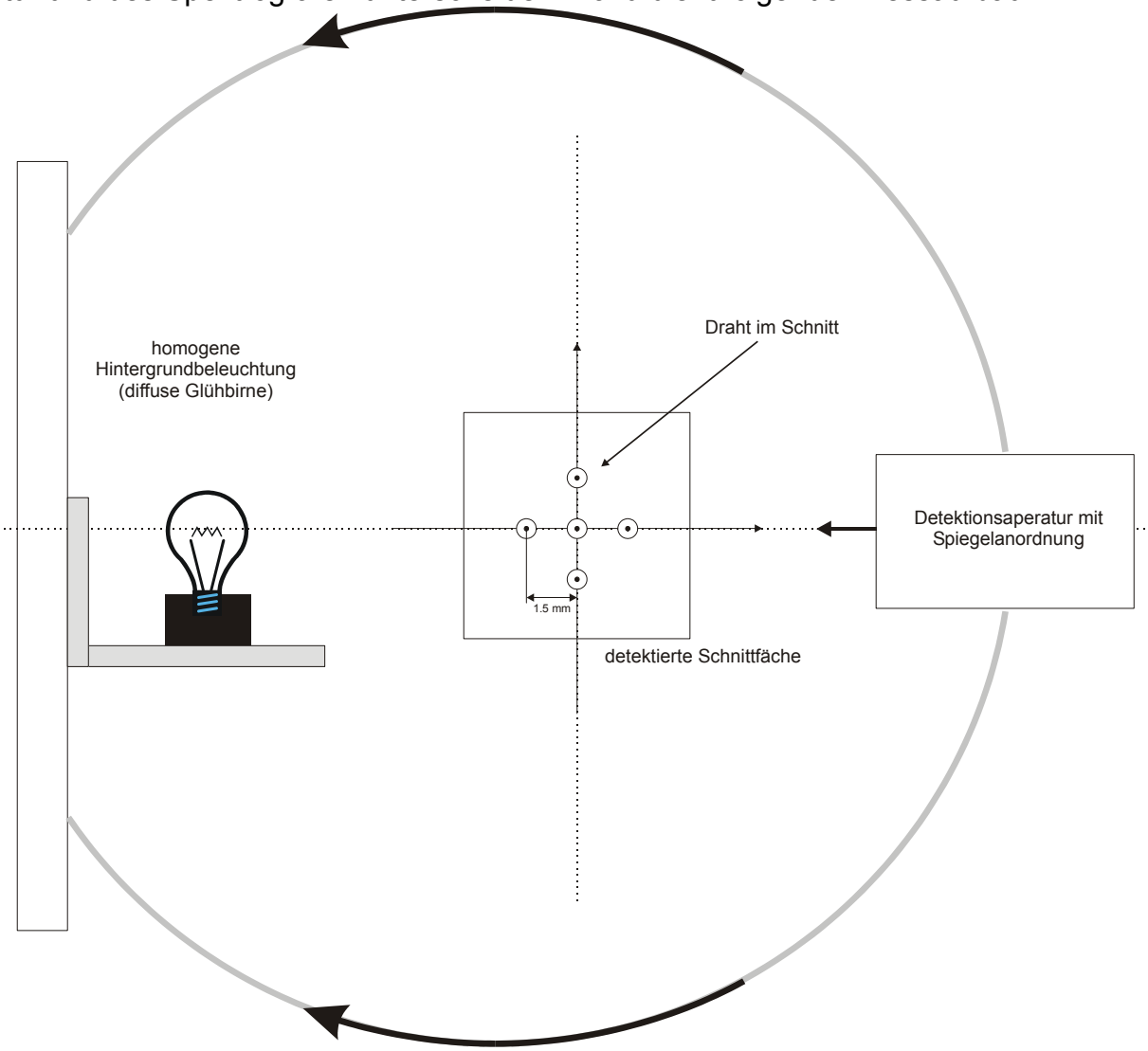


Abbildung 6-1: Versuchsaufbau zur Koordinatenbestimmung

Zur Durchführung der Messung dient ein Draht (Dicke: 0,5 mm) der an fünf unterschiedlichen Positionen gespannt wird. Der Abstand der einzelnen Positionen ist dabei genau 1,5 mm, die mit einem µm – Trieb genau eingestellt werden können. Die Abbildung des Drahtes geschieht durch eine Spiegelanordnung in die optische Detektionseinheit. Die Kamera sieht dabei die Projektion eines Drahteschnittes homogen über alle Spektralbereiche verteilt, da die Glühbirne im sichtbaren Spektralbereich ein annähernd konstantes weißes Spektrum besitzt. Durch eine Invertierung der Aufnahme, kann jetzt eine Messung mit einer „leuchtenden“ Quelle simuliert werden. Um eine möglichst kontrastreiche Messung zu erzielen wird der Auslesebereich hierfür maximal gewählt.
Abbildung 6-2 zeigt links die durchgeführten fünf Messungen. Dabei wurde von der zentralen Position jeweils um 1,5 mm nach rechts und links bzw. nach oben und un-

ten mit dem µm – Trieb verfahren. Anschließend wurden alle Messdaten aufaddiert. Der rechte Bildausschnitt zeigt die rekonstruierten Bilddaten. Diese wurden zuerst einzeln ausgewertet und dann ebenfalls übereinander gelegt.

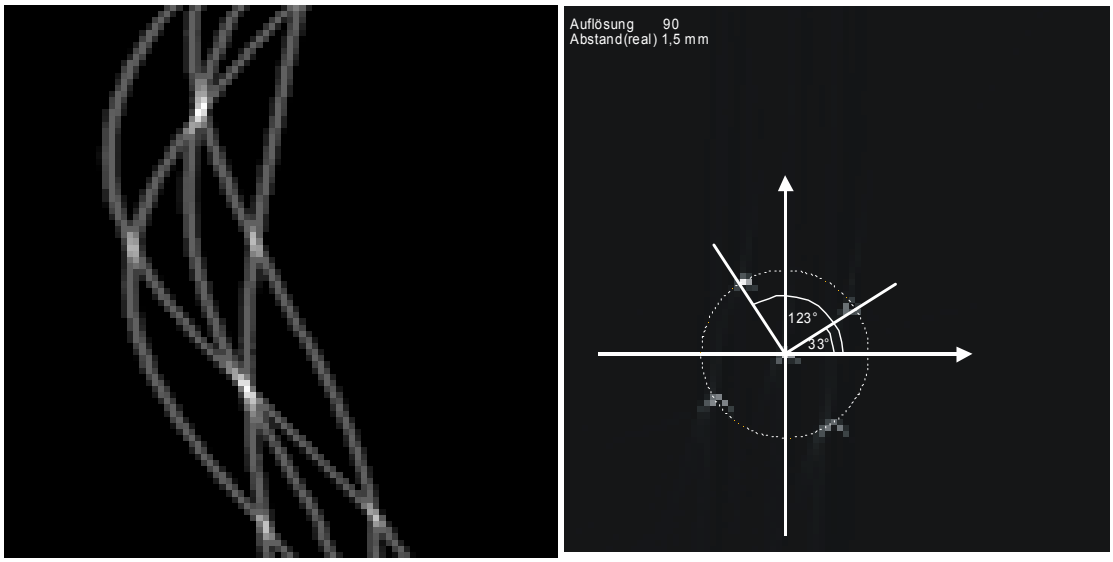


Abbildung 6-2: aufgenommenes (links) und rekonstruiertes (rechts) Bild

An den rekonstruierten Bilddaten lässt sich nun feststellen, dass die Spiegelanordnung das aufgenommene Bild um 33°gegen den Uhrzeigersinn dreht. Dies erleichtert in nun folgenden Experimenten die Zentrierung der Messobjekte. Fährt man die Trommel auf genau 33° bzw. 123° so kann man dort durch Anpassen jeweils der X – bzw. Y – Achse unter zu Hilfenahme der µm – Triebe das Messobjekt sehr schnell zentrieren. Dies ist besonders bei sehr heißen Messobjekten (z.B. Plasmaquellen) notwendig um eine unnötige Erhitzung der Versuchsanordnung zu vermeiden. Auf Grund der Abstände kann man hier schon erkennen, dass die Spiegelanordnung keine Streckung bzw. Stauchung der aufgenommenen Bilder darstellt. Dies soll aber in späteren Messungen noch gezeigt werden.

Übersicht über die Messungen bzw. Messreihen:

Datum	Beschreibung in Stichworten	Seite
28.11.02	Testmessung, sehr schlechte Qualität der Messdaten Vermutung: Fehlzündung bei Start, daraus folgend Toriumverarmung in den Kathoden	125
05.12.02	Messreihe bestehend aus zwei Einzelmessungen: Variation des Kaltgasdurchflusses, bis „kalte Seele“ deutlich sichtbar	125
09.12.02	Messreihe bestehend aus vier Einzelmessungen: Zyklisches Durchtauschen der Kathodenanschlüsse	127
05.03.03	Zwei Messungen mit je zehn Einzelmessungen, bei denen die Plasmabrennercharakteristik auf ihre Temperaturabhängigkeit untersucht werden soll.	131
10.03.03	Eine weitere Messung, die das Ergebnis der Messungen am 05.03.03 unterstreicht.	132
11.03.03	Messung an drei verschiedenen horizontalen Positionen	133

Die einzelnen Messungen werden durch viele äußere Faktoren beeinflusst. Diese können regulierbar sein wie: Kathodenstrom, Kaltgasdurchfluss oder Drall des Plasmagases. Sie können allerdings auch nicht kontrollierbarer Natur sein, wie: Unterschiedliches Abbrennen der Kathoden, statistisches Zustandekommen des Lichtbogenfußpunktes oder Verfälschung aufgrund thermischer Veränderung des Brenners.
Die tomografische Emissionsspektroskopie bietet hier einige hilfreiche Möglichkeiten, diese Effekte zu beurteilen. Um die verschiedenen Messergebnisse deuten zu können, ist es wichtig, das Messobjekt in seiner Funktionsweise zu verstehen. Aus diesem Grund wird hier eine kurze Einführung gegeben:

Einführung in das Messobjekt Mehrkathodenplasmabrenner TRIPLEX

Im Gegensatz zum weit verbreiteten F4-Plasmabrenner besitzt der TRIPLEX-Brenner drei Kathoden.
Dies zeichnet ihn gegenüber den Einkathodenbrennern hinsichtlich verschiedener Punkte aus:

- Stabiles Betriebsverhalten, azimuthale und axiale Instabilitäten des anodischen Lichtbogenfußpunktes treten nicht auf.
- Zeitlich und räumlich stabiler Plasma- Freistrahl

Hier weniger wichtig:

- Daraus folgende geringe Leistungs und Enthalpieschwankungen
- Effektive Pulvererwärmung bei höherem Auftragswirkungsgrad

Aufbau:

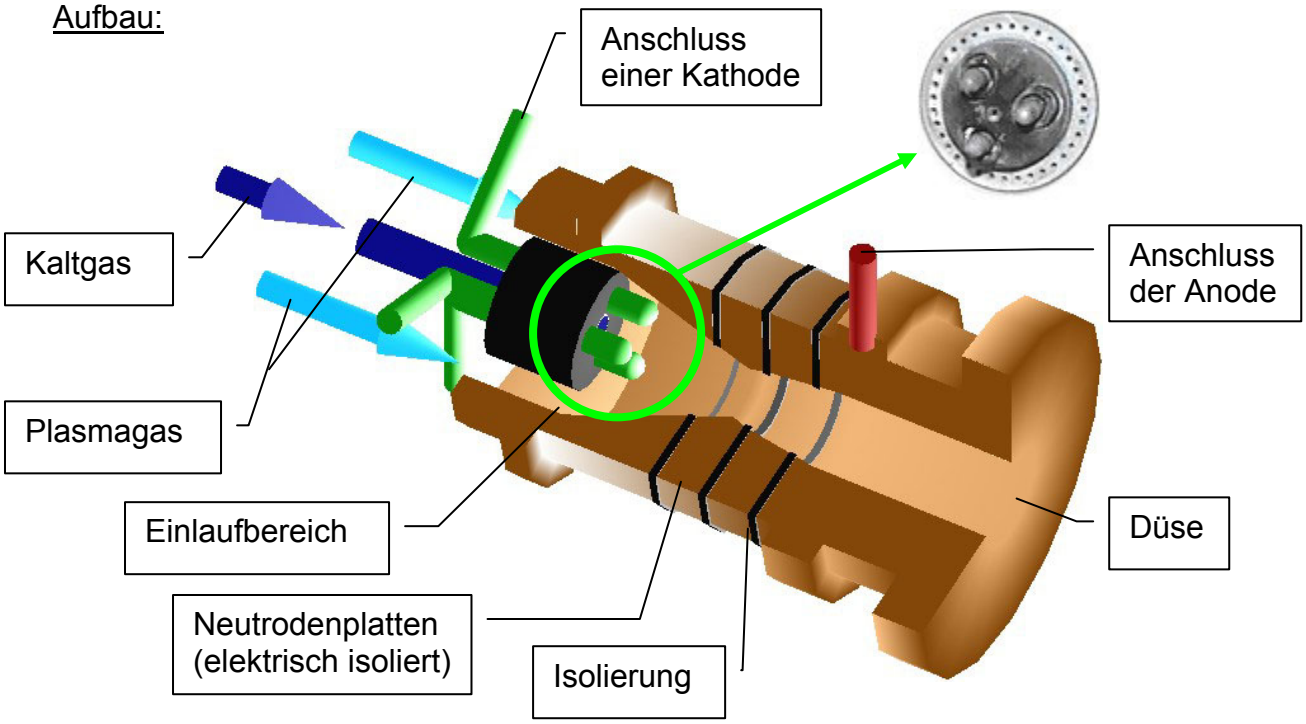
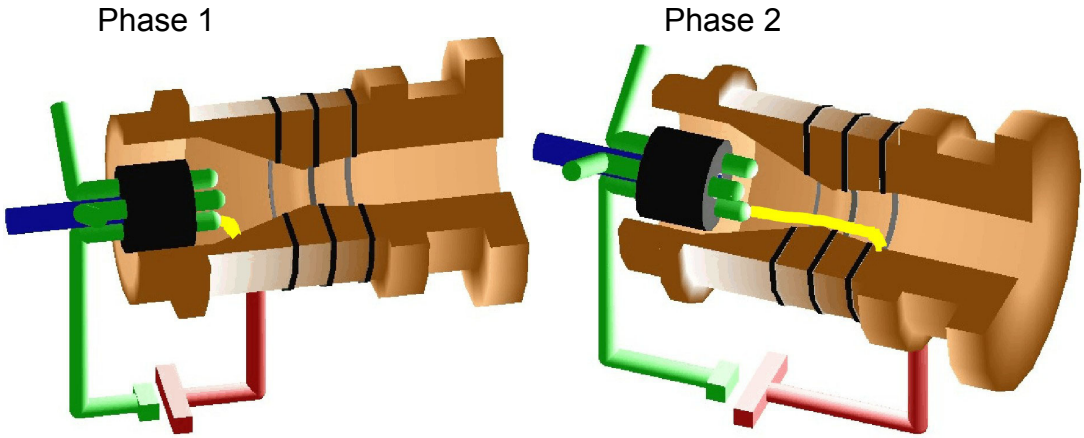


Abbildung 6-3: Das Messobjekt TRIPLEX-Brenner (Aufbau)

Drei elektrisch voneinander isolierten Kathoden mit jeweils eigener Stromeinspeisung ersetzen die einzelne Kathode des F4-Brenners. Der Aufbau der Kathoden entspricht im wesentlichen dem der Einkathodenbrenner. Ihre Durchmesser sind jedoch erheblich kleiner. Wie auch bei den Einkathodenbrennersystemen wird das Gas parallel oder mit Drallwinkel zu den Kathoden in eine rotationssymmetrische Düse eingegeben.
Zusätzlich kann Kaltgas (ebenfalls Argon) über eine Düse zwischen den Kathoden eingeblasen werden, welches die drei Lichtbögen auseinander drückt.

Startvorgang:
Der Brenner kann nicht in einem Schritt in den Betriebszustand überführt werden. Vorbereitend muss Plasmagas (in diesem Fall 33SLPM Argon) strömen. Dann kann die Wasserkühlung zugeschaltet werden (typischer Wert: 6Bar Druck), da sonst Kondenswasser im Inneren des Brenners ein zuverlässiges Starten verhindert. Erst jetzt kann mit dem eigentlichen Startvorgang begonnen werden, der hier in zwei Phasen aufgeteilt ist:



(Bei beiden Grafiken ist zur besseren Veranschaulichung lediglich eine Kathode angeschlossen; in der Realität „brennen“ alle drei Kathoden gleichzeitig.)

Abbildung 6-4: Startphasen des TRIPLEX-Brenner

Phase 1: Zunächst wird der Einlaufbereich an die Stromquelle angeschlossen. Nach einem Stoßstrom von 1500A bildet sich ein leitender Kanal zwischen Einlaufbereich und Kathode.

Phase 2: Das einströmende Plasmagas trägt einige Ladungsträger weiter, so dass sich nach Lösen der Verbindung zwischen Stromquelle und Einlaufbereich und Kontaktieren mit der Anode ein Lichtbogen zwischen Kathoden und Anode ausbildet. Ein Strom von ca. 75A wird dann dauerhaft eingeprägt, was eine Bogenspannung von ca. 150V nach sich zieht.
Das Plasmagas wird so von dem Lichtbogen in den Plasmazustand versetzt und tritt aus der Düsenöffnung aus.

Regulierbar und ausschlaggebend für die hier durchgeführten Messungen sind folgende Parameter:

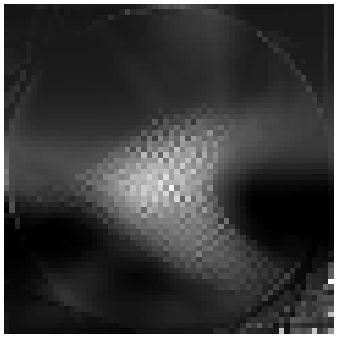
- Kathodenstrom: je höher der Kathodenstrom, um so stabiler ist die Plasmaflamme (gute Ergebnisse bei Gesamtstrom $225A = 3 \cdot 75A$).
- Kaltgasdurchfluss: bei ca. 7SLPM Argon werden die einzelnen Lichtbögen so weit auseinandergetrieben, dass der Einfluss jeder Kathode im Plasmafreistrahl sichtbar wird.
- Betriebsdauer: Gibt die Betriebsdauer des Brenners vor Messungsbeginn an, da es Veränderungen des Plasmastrahls bei wärmer werdendem Brenner geben kann (soll noch untersucht werden).
- Aufgenommene Wellenlänge: wichtig für spätere spektroskopische Auswertungen. Sie wird entweder in [nm] (nur falls Spektraljustage durchgeführt wurde) angegeben oder in Anzahl der Pixel vom linken Bildrand entfernt.

Messung am 28.11.02

Eingestellte Parameter:

Gesamtstrom:	200A
Kaltgasdurchfluss:	3SLPM
Betriebsdauer:	Messung unmittelbar nach Start

Ergebnisse:



Interpretation der Ergebnisse:

Eine Vermutung ist: Nach einer Fehlzündung (durch fehlerhaftes Startkabel) kam es in den Kathoden zu einer Toriumverarmung, was zu Problemen beim Elektronenaustritt führte (Vergrößerung der Austrittsarbeit).
Dies beeinflusste die Stabilität des Brenners immens, die Flamme „flackerte“ deutlich.
Dies ist aber nur eine Vermutung, es kann durchaus noch andere Gründe für die schlechten Messwerte geben. Im aufgenommenen Bild links sind jedenfalls gut Sprünge zu sehen - Es ist selbstverständlich, dass daraus keine brauchbaren Messwerte gewonnen werden können.

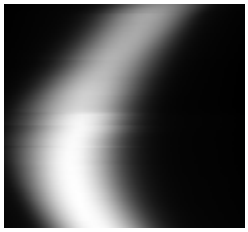
Messreihe am 05.12.02:

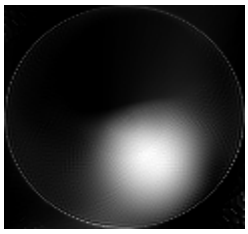
Einzelmessung 1:

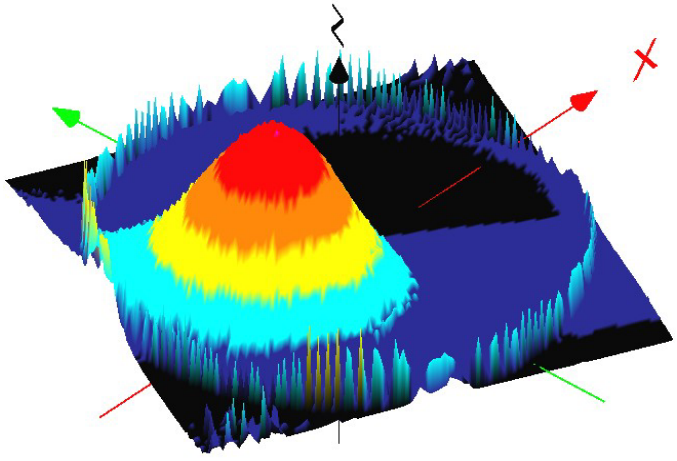
Eingestellte Parameter:

Gesamtstrom:	200A
Kaltgasdurchfluss:	3SLPM
Betriebsdauer:	Messung unmittelbar nach Start

Ergebnisse:







Interpretation der Ergebnisse:

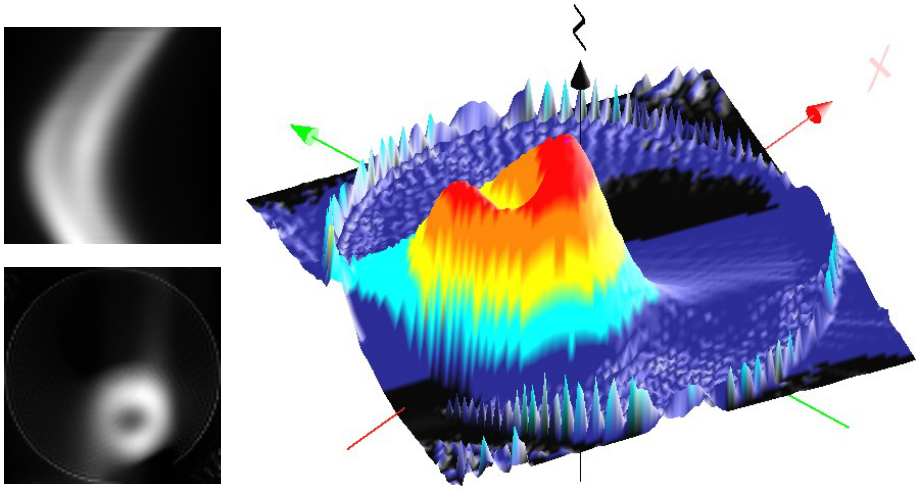
3 SLPM Kaltgasdurchfluss sind zu wenig, um eine deutliche Trennung der drei Lichtbögen im ausgewerteten Bild sehen zu können. Im Vergleich dazu wurde die nächste Messung bei 6SLPM Kaltgasdurchfluss durchgeführt.

Einzelmessung 2:

Eingestellte Parameter:

Gesamtstrom:	225A
Kaltgasdurchfluss:	6SLPM
Betriebsdauer:	5 Minuten nach Start

Ergebnisse:



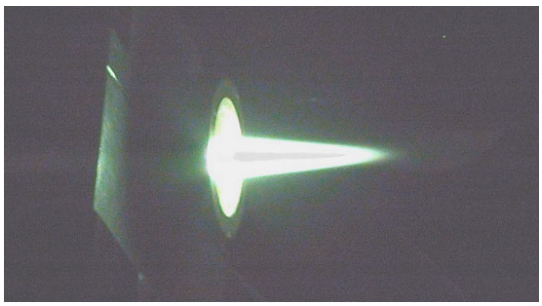


Abbildung 6-5 Fotomontage des Brenners im Betrieb

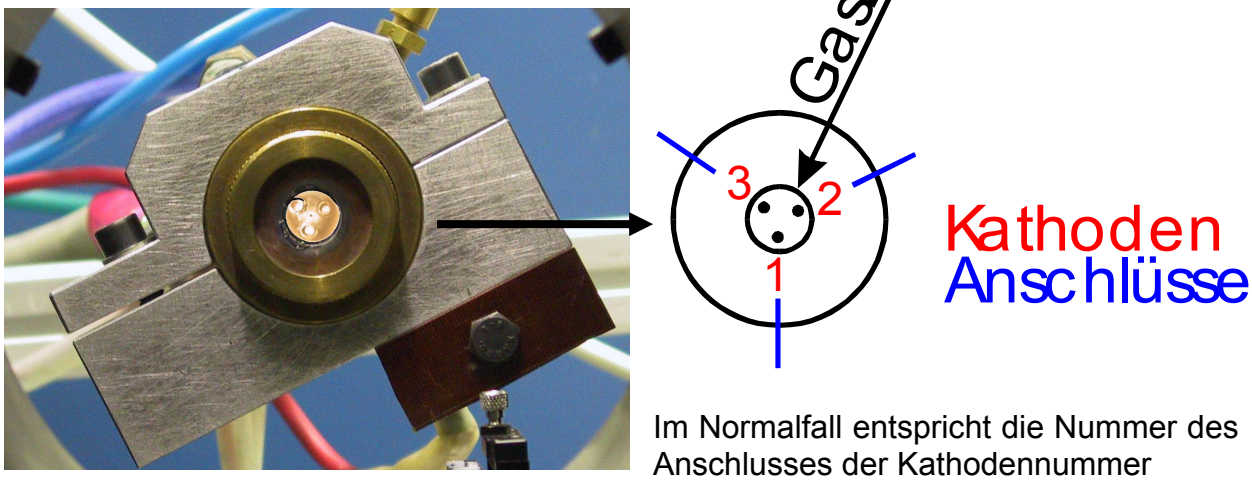
Interpretation der Ergebnisse:

In Abbildung 3 ist schön die „Kalte Seele“ der Plasmaflamme zu sehen. Da mit einer handelsüblichen Kamera emissionsstarke Objekte wie die Plasmaflamme nicht fotografiert werden können, zeigt Abbildung 4 eine Fotomontage des Brenners im Betrieb, so wie ihn das Auge sehen würde.

Messreihe am 9.12.02:

Mit dieser Messreihe sollte herausgefunden werden, ob im Verlauf der Messungen durch Fremdfaktoren (wie Erwärmung des Brenners oder der Versuchsanordnung) Veränderungen in den Messergebnissen sichtbar werden. Aus diesem Grund entsprach die letzte Messung in allen extern regulierbaren Punkten der ersten Messung. Des Weiteren soll die Messreihe zeigen, dass eine unterschiedliche Abnutzung der Kathoden starken Einfluss auf die Plasmaflamme hat. Hierzu wurde zyklisch der Kathodenanschluss durchgetauscht. Effekte durch die externe Verschaltung konnten somit herausgefiltert werden.

Folgende Skizze des Plasmabrenners verdeutlicht die Verschaltung der Kathoden. Eine Tabelle bei jeder Messung zeigt die Kathodenströme, deren Summe aufgrund von Verlusten geringer als der Gesamtstrom ist.



Im Normalfall entspricht die Nummer des Anschlusses der Kathodennummer

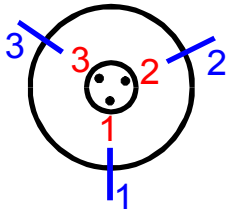
Abbildung 6-6 Bezeichnung der Kathoden und Anschlüsse

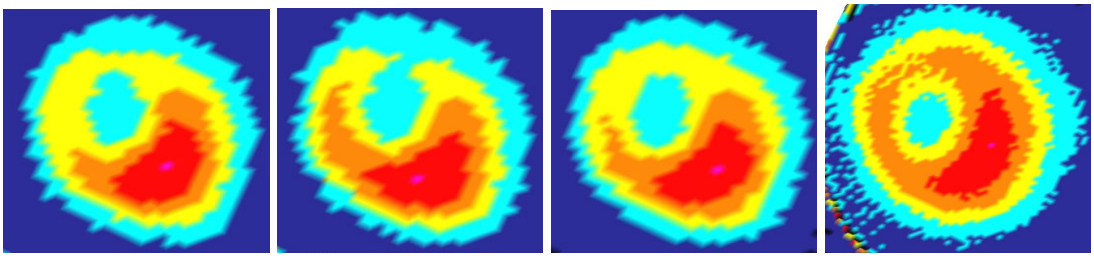
Eingestellte Parameter: Gesamtstrom: 225A
Kaltgasdurchfluss: 7SLPM
Betriebsdauer: siehe bei Einzelmessungen

Messung 1:

Durchlauf 1: Betriebsdauer: 5 Min nach Start, Brenner warm

	Strom [A]	Spannung [V]
Kathode 1	77,9	38
Kathode 2	69,8	42,3
Kathode 3	73,4	39,9

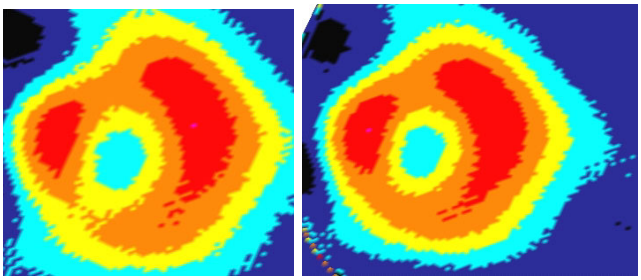




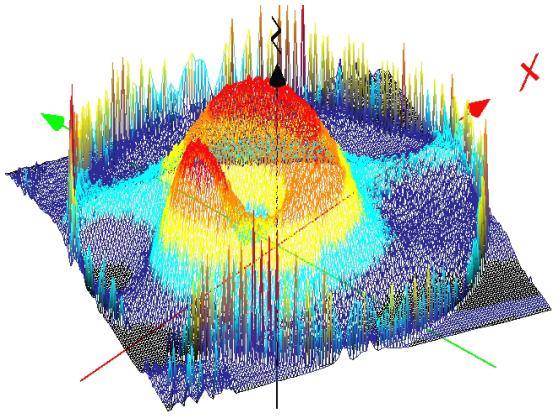
138 Pixel 328 Pixel 516 Pixel 320 Pixel

Durchlauf 2: Betriebsdauer: unmittelbar nach Start, abgekühlt

	Strom [A]	Spannung [V]
Kathode 1	76	39,4
Kathode 2	70,9	42,6
Kathode 3	75,5	39,7



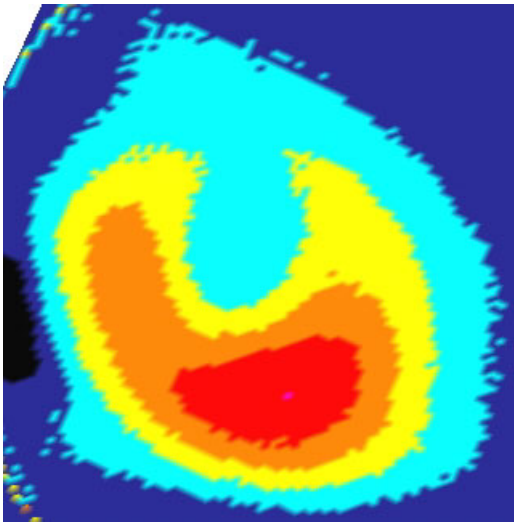
211 Pixel 538 Pixel



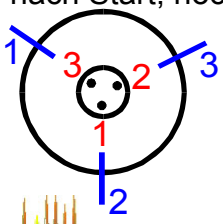
211 Pixel

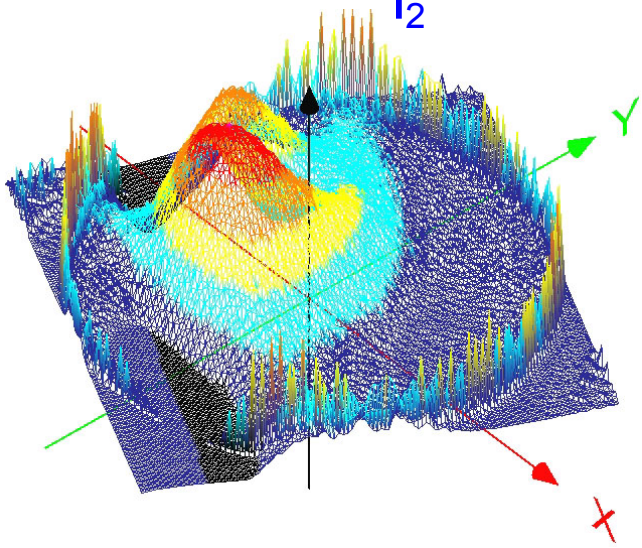
Messung 2: Betriebsdauer: unmittelbar nach Start, noch warm

	Strom [A]	Spannung [V]
Kathode 1	78,3	39
Kathode 2	68,1	42,5
Kathode 3	76,6	38,9



538 Pixel

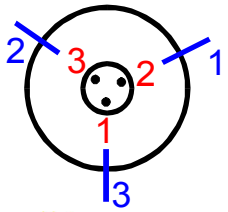


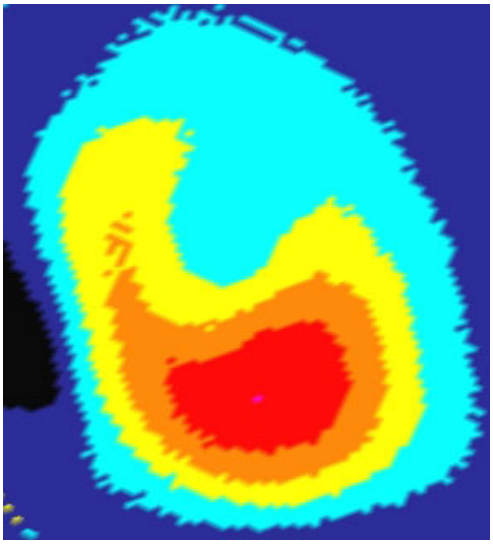


538 Pixel

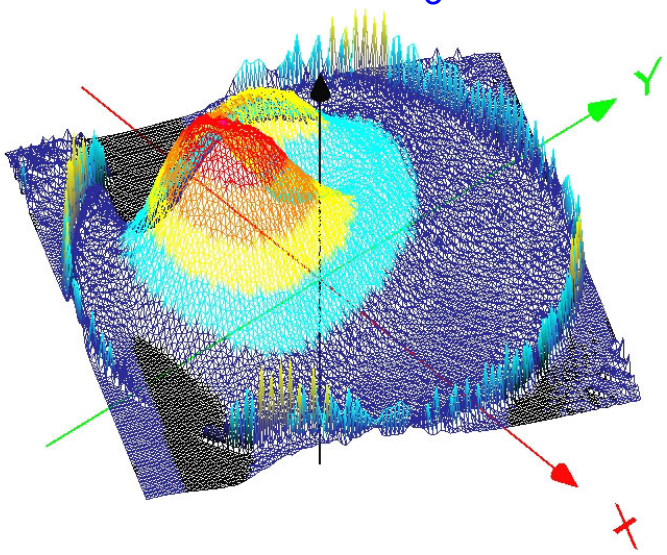
Messung 3: Betriebsdauer: unmittelbar nach Start, noch warm

	Strom [A]	Spannung [V]
Kathode 1	78,4	39,2
Kathode 2	69,3	42,6
Kathode 3	75,1	41,6





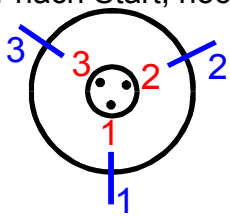
538 Pixel

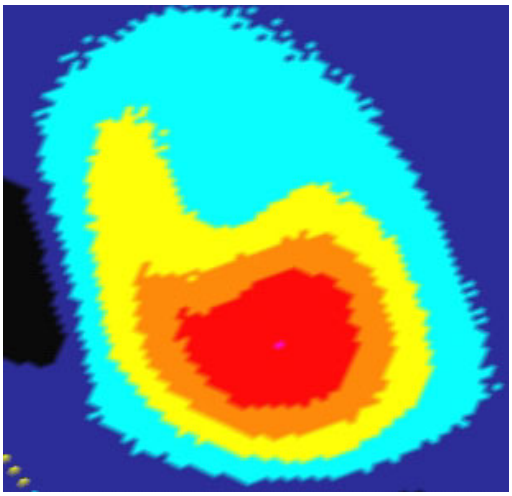


538 Pixel

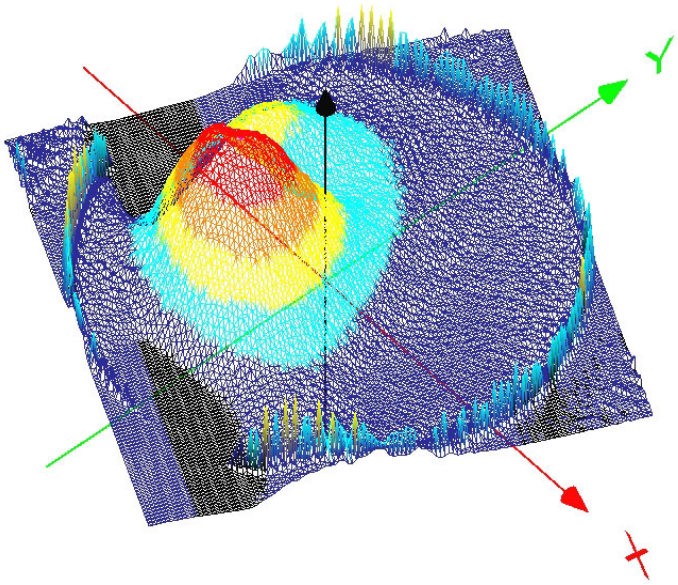
Messung 4: Betriebsdauer: unmittelbar nach Start, noch warm

	Strom [A]	Spannung [V]
Kathode 1	79,5	38,1
Kathode 2	71	42,6
Kathode 3	72,5	41,2





538 Pixel



538 Pixel

Interpretation der Ergebnisse:

Die Erwartung war, dass sich 1. und 4. Messung gleichen, da alle Einstellungen identisch waren. Gerade die aufgenommenen Werte bei der „Wellenlänge“ 538[Pixel] des zweiten Durchlauf der ersten Messung und der vierten Messung sollten sich weitestgehend gleichen.
Dem war jedoch nicht so. Durchlauf 2 der 1. Messung zeigt deutlich zwei Maxima bei den Kathoden 2 und 3, Messung 4 zeigt hingegen ein Maximum bei Kathode 1!
Eine Erklärung hierfür könnte sein:

> Vor der 1. Messung mussten an der Optik einige Einstellungen bei laufendem Brenner durchgeführt werden. Zur eigentlichen Messung war er bereits auf Betriebstemperatur.
> Nach dem 1. Durchlauf wurde der Brenner jedoch abgeschaltet und etwa eine Stunde nicht benutzt, er konnte also vollständig abkühlen.
> Durchlauf 2 wurde dann sofort nach dem Start am kalten Brenner gemessen.
> Für alle weiteren Messungen wurde der Brenner nur für kurze Zeit abgeschaltet, um die Kathoden durchzutauschen, er hatte also nicht genügend Zeit, sich vollständig abzukühlen.
> Durchlauf 2 der 1. Messung ist also als einziger an einem vollständig kalten Brenner gemessen.

Dies könnte ein Indiz dafür sein, dass sich die Charakteristik des Brenners mit dessen Temperatur verändert. Weitere Messungen sollten dahingehend Aufschluss geben.

Der Strom durch Kathode 2 ist stets geringer als der Strom durch die beiden anderen Kathoden. Da eine Änderung der äußeren Beschaltung hier keinen Einfluss hatte, muss dies zwangsläufig an der Kathode selbst liegen.
Die Erklärung hierfür: ein unterschiedlich starkes „Abbrennen“; Kathode zwei ist kürzer, der Abstand zur Anode ist dadurch länger, der elektrische Widerstand also größer.
Tatsächlich sieht man auf den ausgewerteten Bilder, dass stets ein Maximum bei Kathode eins auftritt, die immer „auf maximalem Strom brannte“. Je stärker der Strom von Kathode drei abweicht, umso steiler fällt die Flanke in dieser Richtung ab.

Messreihe am 05.03.03:

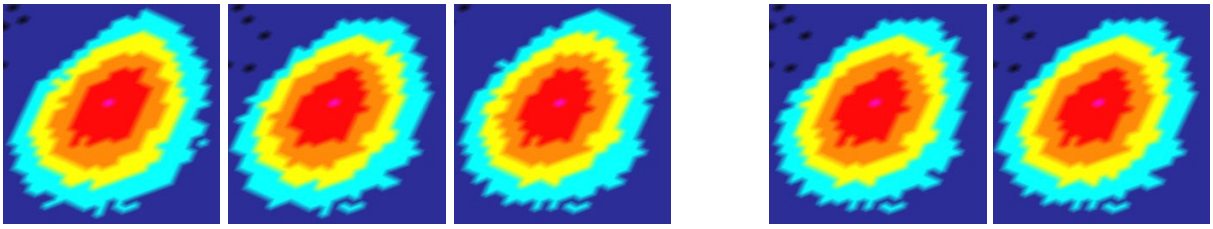
Der Effekt des sich mit der Temperatur verändernden Plasmafreistrahls soll hier insbesondere untersucht werden. Hierfür wurden zwei Messdurchläufe mit jeweils zehn Einzelmessungen durchgeführt. Um möglichst kurze Einzelmessungen zu erhalten wurde mit der niedrigen Auflösung 60 gemessen (Dauer einer Einzelmessung etwa 84 Sekunden)

Messung 1:

Eingestellte Parameter:

Gesamtstrom:	200A
Kaltgasdurchfluss:	3 SLPM
Betriebsdauer:	3 Minuten nach Start

Einzelmessung (Zeit nach Brennerstart)



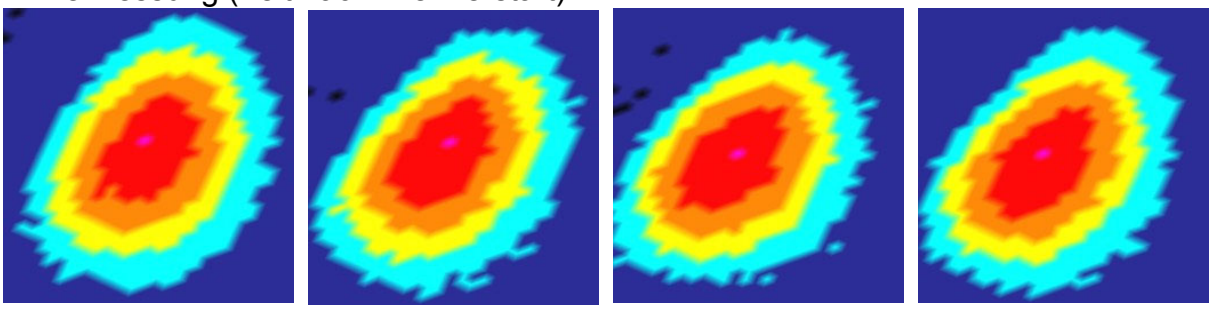
1 (180 sec) 2 (264 sec) 3 (348 sec) … 9 (852 sec) 10 (936 sec)

Messung 2:

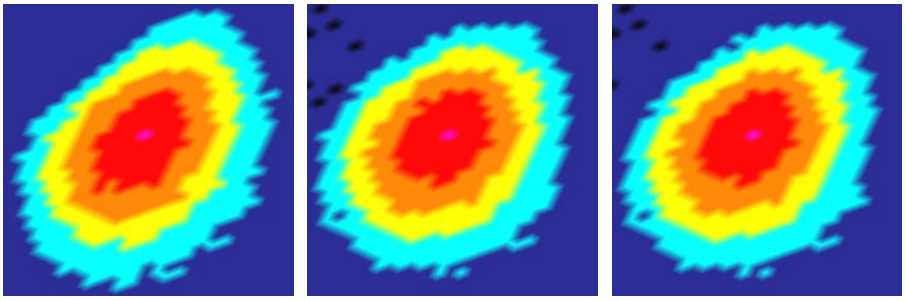
Eingestellte Parameter:

Gesamtstrom:	200A
Kaltgasdurchfluss:	3 SLPM
Betriebsdauer:	sofort nach Start

Einzelmessung (Zeit nach Brennerstart)



1 (0 sec) 2 (84 sec) 3 (168 sec) 4 (252 sec)



5 (336 sec) 6 (420 sec) 7 (504 sec) …

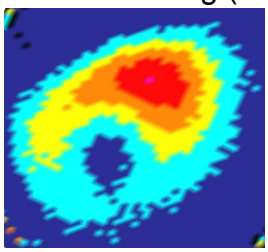
(Keine Veränderung der Flamme ab Einzelmessung 7)

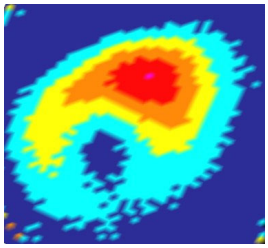
Interpretation der Ergebnisse:
Bei Messung eins stellte sich leider nicht der Effekt ein, der erwartet wurde. Lediglich die ersten drei Einzelmessungen unterscheiden sich ein wenig. Dies könnte daran liegen, dass der Brenner vor Messungsbeginn zu Einstellungszwecken schon drei Minuten lief.
Deshalb wurde eine weitere Messung durchgeführt, diesmal unmittelbar nach Brennerstart. Hier ist eher - wenn auch sehr gering – eine Veränderung des Plasmafreistrahls zu beobachten.

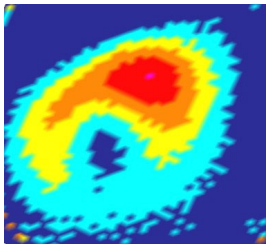
Messung am 10.03.03:
In Anlehnung an die Messungen am 05.03.03 sollte diese Messung den Effekt des sich mit der Brennertemperatur verändernden Plasmafreistrahls deutlicher zeigen.
Hierzu wurden alle Einstellungen übernommen, bis auf den Durchfluss des Kaltgases der auf 7SLPM eingestellt wurde.

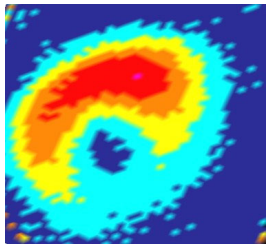
Eingestellte Parameter:	Gesamtstrom:	200A
	Kaltgasdurchfluss:	7 SLPM
	Betriebsdauer:	sofort nach Start
	Aufgenommene Wellenlänge:	600[nm]

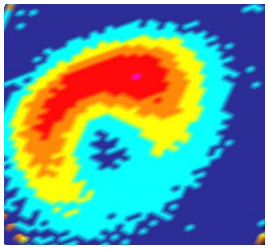
Einzelmessung (Zeit nach Brennerstart)

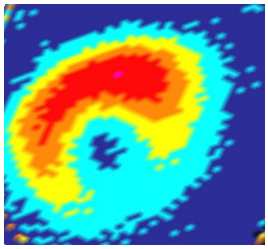

1 (0 sec)

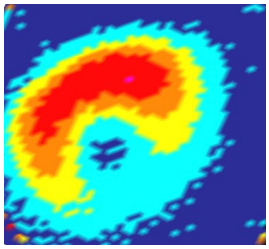

2 (84 sec)

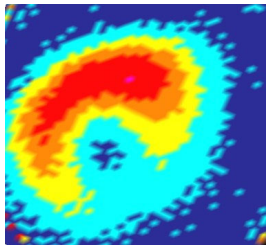

3 (168 sec)

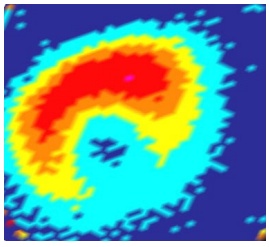

4 (252 sec)

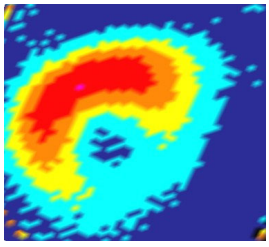

5 (336 sec)


6 (420 sec)


7 (504 sec)


8 (588 sec)


9 (672 sec)


10 (756 sec)

Interpretation der Ergebnisse:
Nun ist sehr deutlich zu erkennen, dass sich der Plasmafreistrahl mit zunehmender Temperatur stark ändert.
Eine Vermutung ist, dass sich die Anodenfußpunkte der drei Lichtbögen nicht symmetrisch auf der Anode einstellen und abhängig von der Temperatur des Brenners wandern. Deshalb konnte man bei den Messungen vom 05.03.03 nur wenig erkennen, da die Einflüsse jedes Lichtbogens bei geringem Kalgasdurchfluss zu nur einem Maximum verschmelzen.

Messreihe am 11.03.03:

Diese Messung nimmt den Plasmastrahl des Triplexbrenners an drei verschiedenen horizontalen Positionen auf. Es werden dabei alle Bilder gespeichert um eine Aussage über die räumliche Form des Plasmafreistrahls zu erhalten.

Eingestellte Parameter

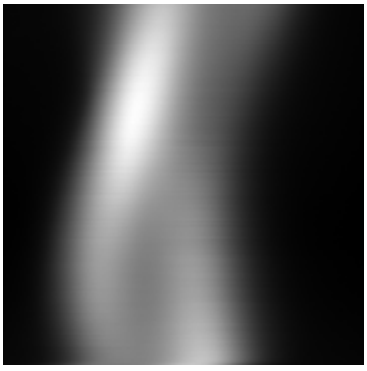
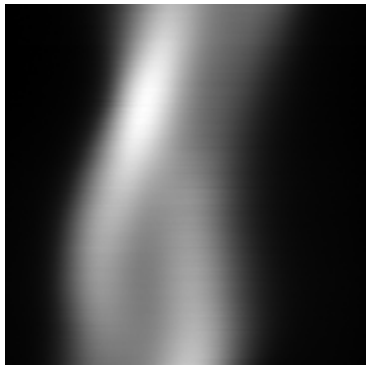
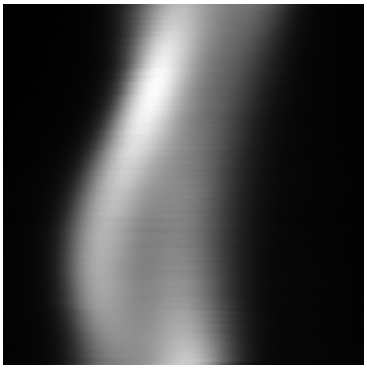
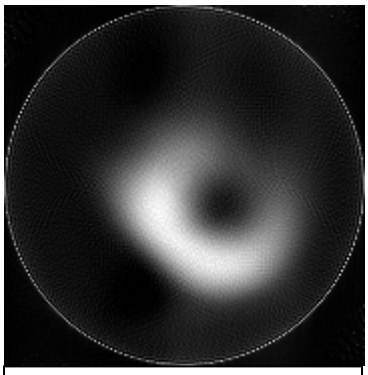
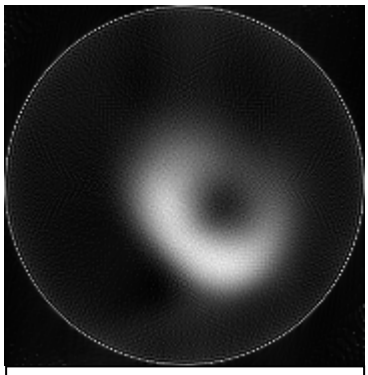
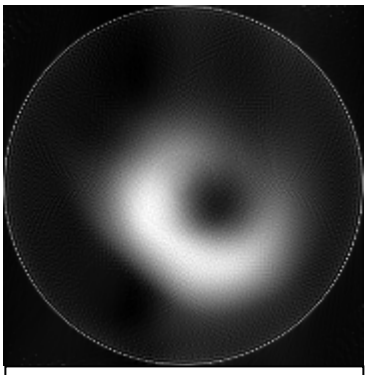
Gesamtstrom:	200A
Kaltgasdurchfluss:	7 SLPM
Betriebsdauer:	ca. 15 Minuten nach Brennerstart

Einstellung der Aufnahme:

Auflösung:	180 Schritte
Aufgenommenen Linien:	Aufzeichnung gesamte Messung

Einstellungen der Auswertung:

Auflösung:	180
Linienbreite:	14
Summierte Zeilen:	2

nahe am Plasmabrenner	5 mm entfernt	10 mm entfernt

Deutung der Ergebnisse:

Wie in vorherigen Messungen gezeigt wurde ist hier keine Instabilität auf Grund der Veränderung des Plasmabrenners zu erwarten. Auch die Messapparatur wurde ausreichend gekühlt und geht ebenfalls nur als konstanter Faktor ein. Die Messungen sind somit direkt vergleichbar.

Die Bilder zeigen optisch nur geringe Unterschiede. Dies liegt an der Normierung der dargestellten Bilder.
Trägt man die Messwerte spektral auf und normiert ebenfalls auf die stärkste Spektrallinie so zeigen sich kleine Unterschiede. Die jeweiligen ausgewerteten Messwerte bei gleicher Wellenlänge verändern sich ohne erkennbaren Zusammenhang. Zur genaueren Bestimmung sind allerdings noch weitere Messungen notwendig.

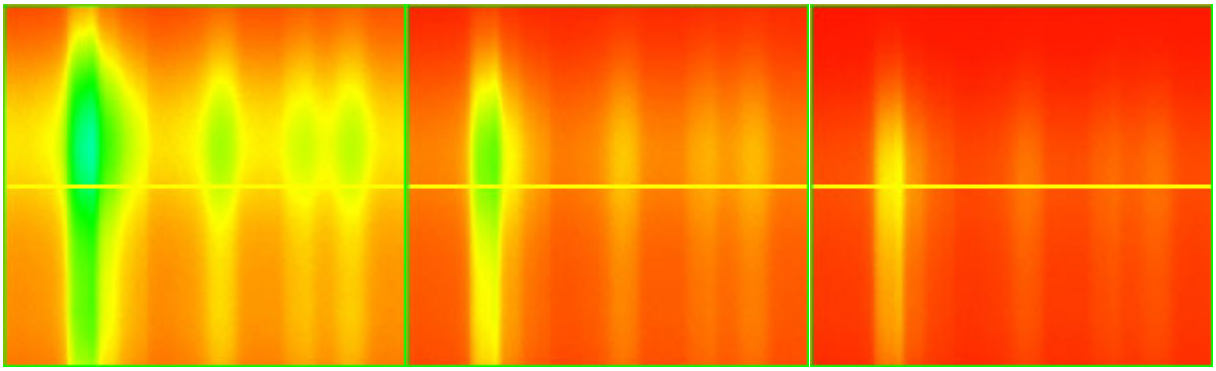


Abbildung 6-7: Falschfarbendarstellung der aufgenommenen Bilder (unnormiert von links nach rechts: direkt am Brennerausgang, 5 mm entfernt, 10 mm entfernt)

Betrachtet man die aufgenommenen Bilder (unnormierte) zeigen sich deutliche Intensitätsunterschiede. Die Intensität der aufgenommenen Bilder nimmt wie erwartet mit der Entfernung vom Brenner ab (vgl. Abbildung 6-7).

7 Zusammenfassung

In den vergangenen Jahren wurden am Institut - im Verlauf mehrerer Diplomarbeiten - die Komponenten eines Meßaufbaus zur tomografischen Emissionsspektroskopie an thermischen Plasmastrahlen entwickelt und aufgebaut. Die wesentliche Aufgabe der vorliegenden Diplomarbeit bestand nun darin, diese Einzelkomponenten zu einem homogenen Meßsystem zusammenzufassen und erste Meßreihen am Rekombinationsplasmastrahl eines Plasmaspritzbrenners durchzuführen.
Hauptbestandteil der Arbeit war die Entwicklung eines Softwarepakets zur Experimentsteuerung, Datenaufnahme, Meßdatenauswertung und Ergebnispräsentation. Außerdem wurden Komponenten zur spektralen Kalibrierung, zur Meßdatensimulation und Experimentvorbereitung implementiert. Die Software ist in DELPHI 6 Syntax für Windows 2000 erstellt und genügt damit den Anforderungen an moderne Softwareprojekte bezüglich Bedienungskomfort, Modularität und Erweiterbarkeit. Besondere Anforderungen bestanden in der sicheren Ansteuerung peripherer Geräte (neues Kamerasystem, Schrittmotoren-Interface) und der strukturierten Organisation von Meßdaten, Ergebnissen und zugehörigen Experimenteinstellungen. (Die bei bildgebenden Experimenten anfallenden Datenmengen sind beachtlich; im vorliegenden Fall bis zu 500MB je Messung). Das entwickelte Software-Programm bietet über eine komfortable und übersichtliche grafische Oberfläche Zugriff auf alle einstellbaren Experimentkomponenten; der eigentliche Meßablauf ist vollautomatisiert und wird vom Rechner überwacht. Im Präsentationsteil besteht Zugriff auf alle durchgeführten Messungen, wobei die Darstellung der Ergebnisse mittels OpenGL dreidimensional erfolgt.
Auch an der bestehenden Experiment-Hardware wurden einige Veränderungen vorgenommen. Neben konstruktiven Verbesserungen (neue Wellenkupplung für Drehmechanismus, Einbau eines Kühlsystems für stationären Meßbetrieb) und kleineren Änderungen in der Optik wurde vor allem der bisher provisorisch eingesetzte Bildaufnehmer durch eine neue CCD-Kamera ersetzt, die jetzt fest mit dem Spektrografen verbunden ist.
Schließlich konnten noch – neben einer Vielzahl von Kalibriermessungen an Spektrallampen und anderen Objekten – eine ganze Reihe von Messungen am Plasmastrahl eines Plasmaspritzbrenners durchgeführt werden. Die Ergebnisse dieser Messungen sind vielversprechend; es konnten reproduzierbar charakteristische Strukturen im Plasmafreistrahl eines Brenners vom Typ TRIPLEX nachgewiesen werden. Bezüglich der spektroskopischen Auswertung der Ergebnisse zur Plasmatemperaturbestimmung sind im Software-Projekt bereits alle wesentlichen Komponenten enthalten. Die dafür benötigten umfangreichen Meßreihen können mit dem jetzt verfügbaren Meßaufbau problemlos durchgeführt werden.

8 Literaturverzeichnis

[1] Pock, S., Bestimmung der Intensitätsverteilung von Plasmastrahlen mittels Computertomographie, Diplomarbeit, Universität der Bundeswehr München, 1997.

[2] Schwinn, R., Weiterentwicklung von Rechnerprogrammen der Plasmadiagnostik, Diplomarbeit, Universität der Bundeswehr München, 1998.

[3] Schindler, N., Entwurf und Aufbau eines Meßplatzes zur tomographischen Diagnostik an Plasmastrahlen, Diplomarbeit, Universität der Bundeswehr München, 1998.

[4] Jochen, P.S., Mark, Realisierung der Optik und Entwicklung der Software zur Steuerung des Meßplatzes, Diplomarbeit, Universität der Bundeswehr München, 1999.

[5] Kohler, Bernd, Untersuchungen am Plasmastrahl mittels Emissionsspektroskopie, Diplomarbeit, Universität der Bundeswehr München, 1989.

[6] Bappert, Andreas, Tomografische Bildverarbeitung zur lokalen Bestimmung der Emissionskoeffizienten von Plasmastrahlen, Diplomarbeit, Universität der Bundeswehr München, 1995.

[7] Ahoyo, Maxime, Tomographische Emissionsspektroskopie - Inbetriebnahme eines neuentwickelten Versuchsstandes, Diplomarbeit, Universität der Bundeswehr München, 1999.

[8] Zierhut, Jochen, Entwicklung von Diagnostikverfahren zur Optimierung von Plasmaspritzsystemen, Dissertation, Universität der Bundeswehr München, 2000.

[9] isert-electronic, Schrittmotorantrieb / Interfacekarte 4.0, Bedienungsanleitung, Eiterfeld, 1988.

[10] SOLAR TII, SL 100 Lens Spectrograph, User´s Manual, Minsk, 1996.

[11] Data Translation UM-14358-A, MACH Series DT3152, Bedienungsanleitung.

[12] Haslbeck, Peter., Entwicklung eines Dreikathoden – Plasmaspritzbrenners unter Anwendung adaptiver plasmadiagnostischer Methoden, Dissertation, Universität der Bundeswehr München, 1995.

[13] Hartmann, R., Entwicklung eines Breitstrahl-Plasmabrenners zur Behandlung planarer Oberflächen, Dissertation, Universität der Bundeswehr München, 2000.

[14] National Institute of Standards and Technology (http://physlab2.nist.gov), 100 Bureau Drive, Stop 8400, Gaithersburg, Maryland 20899-8400.

[15] Fischer, Wolgang, Flammenanalyse mittels tomografischer und multispektraler Bildverarbeitung, Dissertation, TU Hamburg – Harburg 1992.

[16] Forster, G., Bestimmung von Parametern thermischer Plasmen mittels Thomsonstreuung, Dissertation, Universität der Bundeswehr München, 1995.

[17] Mayr, W., Bestimmung der lokalen Geschwindigkeits- und Größenverteilungen von Partikeln im Plasmastrahl mittels Laser-Doppler-Anemometrie, Dissertation, Universität der Bundeswehr München, 1983.

[18] Reusch, A., Die Entwicklung eines Laser-Doppler-Meßsystems und seine Anwendung bei Verfahren des thermischen Beschichtens, Dissertation, Universität der Bundeswehr München, 1995.

[19] Streibl, T., Entwicklung eines bildgebenden Partikeldiagnostikverfahrens zur Untersuchung Thermischer Spritzprozesse, Dissertation, Universität der Bundeswehr München, 2002.

[20] Wagner-Martens, A., Hardwareerstellung zur Automatisierung des Plasmabrennerbetriebes mit einer speicherprogrammierbaren Steuerung (SPS), Diplomarbeit, Universität der Bundeswehr München, 2002.

[21] Doubeck, P., Softwareerstellung zur Automatisierung des Plasmabrennerbetriebes mit einer speicherprogrammierbaren Steuerung (SPS), Diplomarbeit, Universität der Bundeswehr München, 2002.

[22] Payling, R., Optical emission lines of the elements, Wiley, 0-471-62378-4, Chichester, 2000.

Abgrenzung der einzelnen Arbeiten:

Eine klare Abgrenzung der einzelnen Arbeiten ist nur schwer möglich. Lediglich das Softwareprojekt lässt auf Grund seines modularen Aufbaus eine Trennung zu.

Aufteilung des Softwareprojektes:

Folgende Programmteile wurden im Wesentlichen erstellt von Björn Marlow:
- Schrittmotorsteuerung
- Drehachsenbestimmung
- Spektrale Justage
- Grafische Überwachung
- Simulation tomografischer Messdaten
- 3D – Browser
- Messdatenaufarbeitung
- Checklist

Folgende Programmteile wurden im Wesentlichen erstellt von Andreas Kraft:
- Livebild
- Aufnahme
- Bildbereich auswählen
- Messung
- Tomografische Auswertung
- Serienbildbrowser
- Spektroskopische Auswertung
- Boltzmann – Plot
- HTML – Konverter
- Messwerte addieren